Visualisation of Structured Data through Generative Probabilistic Modeling

by

Nikolaos Gianniotis

A thesis submitted to

The University of Birmingham

for the degree of

DOCTOR OF PHILOSOPHY

School of Computer Science

The University of Birmingham

September 2007

Abstract

This thesis is concerned with the construction of topographic maps of structured data. A probabilistic generative model-based approach is taken, inspired by the GTM algorithm. Depending on the data at hand, the form of a probabilistic generative model is specified that is appropriate for modelling the probability density of the data. A mixture of such models is formulated which is topographically constrained on a low-dimensional latent space. By constrained, we mean that each point in the latent space determines the parameters of one model via a smooth non-linear mapping; by topographic, we mean that neighbouring latent points generate similar parameters which address statistically similar models. The constrained mixture is trained to model the density of the structured data. A map is constructed by projecting each data item to a location on the latent space where the local latent points are associated with models that express a high probability of having generated the particular data item.

We present three formulations for constructing topographic maps of structured data. Two of them are concerned with tree-structured data and employ hidden Markov trees and Markov trees as probabilistic generative models. The third approach is concerned with astronomical light curves from eclipsing binary stars and employs a physical-based model. The formulation of the all three models is accompanied by experiments and analysis of the resulting topographic maps.

To my parents Μαρία and Σταμάτης

Acknowledgements

I would like to express my great appreciation to my supervisor Dr. Peter Tiňo not only for steering my PhD to interesting directions with thoughtful guidance, but also for inspiring me with excitement to pursue further research in the domain of machine learning in the future.

I would also like to thank Steven Spreckley, Dr. Somak Raychaudhury and Dr. Ian Stevens from the School of Physics and Astronomy, University of Birmingham, for their collaboration and provision of the astronomical model, dataset of light curves and prior distributions in the work presented in chapter 6.

Finally, I could not have completed my PhD studies without the studentship granted by the School of Computer Science, for which I am grateful.

Contents

List of Figures

List of Tables

Notation and Abbreviations

A list of the most repeated notation and symbols used in the thesis follows.

Γ	Non-linear RBF mapping from space $\mathcal{V}$ to data/model space
$\delta(.)$	Dirac impulse function
$\delta_{i,j}$	Kronecker delta
$\mathcal{D}$	Dataset
$D_{KL}[.\|\|.]$	Kullback-Leibler divergence
$E_x[.]$	Expectation operator with respect to distribution of variable x
$\boldsymbol{\theta}$	Parameter vector of probabilistic model
$\boldsymbol{\Theta}$	Parameter vector of mixture model
$\mathcal{H}$	Space of all models $p(\cdot\|\boldsymbol{x})$
$\mathcal{L}$	Model likelihood
$\mathcal{M}$	Two-dimensional manifold of noise models $p(\cdot\|\boldsymbol{x})$ constrained on latent space $\mathcal{V}$
$\mathcal{N}$	Gaussian distribution
$\boldsymbol{O}$	Light curve from an eclipsing binary system
$p(\cdot\|\boldsymbol{x})$	Local noise model addressed by latent point $\boldsymbol{x}$
$\boldsymbol{s}$	Sequence data item
$\boldsymbol{t}$	Vector data item
$\mathcal{V}$	Latent space embedded in space $\mathcal{H}$
$\boldsymbol{x}$	Coordinate vector of neuron on lattice (in neural-based formulations); Latent point (in GTM formulations)
$\boldsymbol{y}$	Tree data item
$\boldsymbol{W}$	Weight matrix
$\boldsymbol{W}_i$	i-th row of weight matrix
$\mathcal{Z}$	Set of indicator-hidden variables

EM	Expectation-maximisation (algorithm)
FIM	Fisher information matrix
GTM	Generative topographic map (algorithm)
HMM	Hidden Markov model
HMTM	Hidden Markov tree model
KLD	Kullback-Leibler divergence
MTM	Markov tree model
RBF	Radial basis function
SOM	Self-organising map
SOMSD	SOM for structured data
TPB	Traffic-policemen benchmark

Chapter 1

Introduction

1.1 Topographic Mapping

Topographic mapping [Kohonen, 1990, Hammer et al., 2004, Svensén, 1998, Kabán and Girolami, 2001] is the data processing technique of constructing maps that capture relationships between the data items in a given dataset. In geographical maps the affinities between objects on the map are in correspondence to the distances between the real world objects that are represented on the map. In topographic maps of data, distances between data reflect the similarity/closeness of the data items which may be Euclidean, Mahalanobis, statistical etc. Clearly topographic mapping is a data visualisation technique when the constructed map is a two- or three-dimensional map, but the term data visualisation[1] spans a much wider thematic area than topographic mapping. In [Keim and Ward, 2002] a wide collection of data visualisation methods is presented, from techniques that enhance the presentation of data by geometrically transforming displays to techniques that produce plots capable of interaction, zooming or dynamic projection. For example in [Kleiberg et al., 2001] an approach to visualising data structures is presented that is based on the adeptness of the human visual system of observing large numbers of branches and leaves on a botanical tree. The approach is demonstrated on a file system by adopting a botanical representation where files, directories etc. are represented by elements such as branches or leaves.

In this work we are not concerned with representational issues such as the adoption of suitable colour schemes, icons or graphics that consider particular aspects of the human visual system when constructing topographic maps. In our construction of a map, data items are simply represented as points. The crux of the work is to endow topographic maps with a clear

[1] Nevertheless, after this clarification we shall interchangeably use the term "visualisation" instead of the more accurate "topographic mapping".

understanding of *why* data points are mapped to their particular locations and *how* to interpret the distances between them. The data items that will concern us here are structured types. Specifically, a substantial part of this work is concerned with tree-structures. Moreover, a real world example is studied at the penultimate chapter of the thesis, where a particular data type holding information on a physical system is accommodated in the construction of topographic maps.

1.2 Two Approaches to Topographic Mapping

Topographic mapping is a valuable tool for the analysis and interpretation of multivariate data. The Self-Organising Map (SOM) [Kohonen, 1990] is one of the most celebrated tools that is of vast assistance to this task. SOM is a type of neural network that allows a nonlinear projection of data residing in a high dimensional space to a lower dimensional projection space. The lower dimensional space is a discrete lattice of neurons (for visualisation purposes a two dimensional lattice). The impact of SOM has been of great magnitude and it has established a kind of paradigm that a number of techniques have followed. According to the SOM paradigm, the formation of the map is realised by iterating two steps of competition and cooperation among the neurons. The competition step involves the presentation of an input pattern and calculation of the response of all neurons. The neuron with the greatest response is declared the winner of the competition. In the cooperation step, the winning neuron is appropriately adjusted to increase its future response to that particular pattern. Moreover, neurons that belong to the neighbourhood (on the lattice) of the winner are also adjusted to increase their future response, albeit proportionally to an (usually) exponentially decaying distance from the winner. Techniques that belong to this paradigm typically modify SOM by equipping neurons with additional feed-back connections that allow for natural processing of recursive data types such as sequences or trees. Typical examples of such recursive neural-based models are the Temporal Kohonen Map [Chappell and Taylor, 1993], recurrent SOM [Varsta et al., 1997], recursive SOM [Voegtlin, 2002], merge SOM [Strickert and Hammer, 2005] and SOM for structured data [Hagenbuchner et al., 2003].

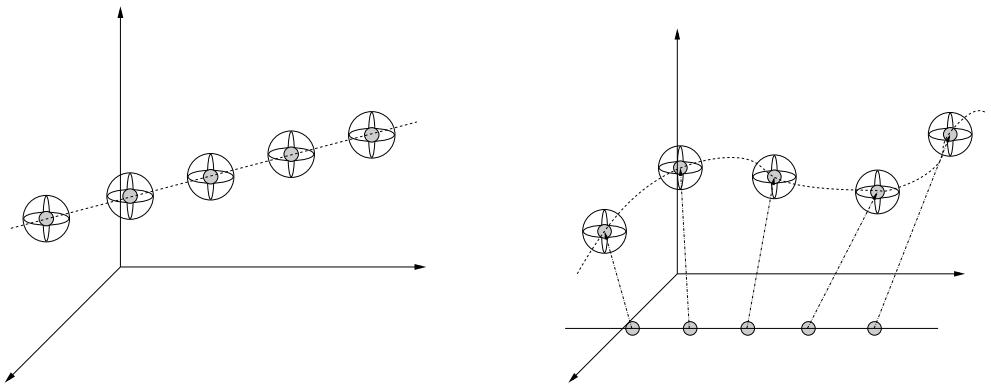
Nevertheless, the heuristic nature of SOM inherently brings about certain limitations, for example the lack of a principled cost function (although see developments in e.g. [Heskes, 1999][2]). Comparison of map formations resulting from different initialisations, parameter settings or

[2]Heskes [Heskes, 1999] suggests a modified version of SOM by redefining the codebook vector (winner unit) associated with an input as the one closest to the input with respect to *averaged* distance across its local neighbourhood on the codebook lattice.

optimisation algorithms can be problematic. The aforementioned recursive neural-based models also inherit such problems from SOM. Again, formulation of a principled cost function is problematic (although see developments in [Hammer et al., 2004] along the lines of [Heskes, 1999]). Also problematic is the explanatory interpretation of the visualisation results in such approaches. Clusters may be formed on the map that indicate some close relationship between the concerned structured data items, but there is no explanation on what the characteristics of the clusters are. Of course one can inspect the individual data points to deduce those relationships once the map has been formed, but reasoning about mapping of new data items (not used for model fitting) can be still challenging.

The Generative Topographic Map (GTM) algorithm [Bishop et al., 1998] was introduced as a principled probabilistic analog to SOM. As a generative model GTM realises a "noisy" low dimensional manifold in a high dimensional data space. It can be used to model a given training dataset by adjusting its parameters so that the model-generated data lying around the noisy low dimensional manifold match (in the distribution sense) the training data. GTM is a mixture of local generative models (spherical Gaussians) that adheres to topological constraints (constraints on the values that means of the Gaussians can take). A simple example is that of requiring that the means belong to a straight line. This could be useful if we believed that the data are intrinsically one-dimensional and are adequately represented by a "noisy line". This situation is illustrated in Fig. 1.1(a). The line on which the means of local Gaussians are placed can be viewed as an image of a one-dimensional interval under a linear (affine) map. Alternatively, one may want to constrain the Gaussian means to lie on a smooth curve. In that case, the one-dimensional interval would be embedded in the high dimensional data space through a smooth non-linear mapping. This is illustrated in Fig. 1.1(b). The GTM belongs to the class of so called latent-variable models with latent space being the one-dimensional interval through which the Gaussian means are constrained.

GTM sets a paradigm of a generative probabilistic approach to the construction of topographic maps. In this work we extend this paradigm to the visualisation of structured data. A substantial part of this work is concerned with developing an extension for tree-structured data that employs hidden Markov tree models as noise models. We compare our approach with a candidate member of the recursive neural-based techniques, the SOM for structured data (SOMSD) [Hagenbuchner et al., 2003]. This comparison concerns also recursive neural-based approaches in general and serves the purpose of illustrating the benefits of a principled probabilistic model-based formulation. For example, the generative nature of our model formulation provides us with an explanatory insight as to how the data might have been generated. By observing the



(a) (b)

Fig. 1.1: Spherical Gaussians constrained on a one-dimensional line (a), spherical Gaussians constrained on a one-dimensional curve (b). Note that the straight line (latent space) in (b) does not belong to the data space and is only plotted in the same figure as its image for convenience.

generative process and its parameters we can understand characteristics of clusters of projected data items and/or discern other patterns in the data. Also, the smooth character of the embedding map from the latent space into the model space enables us to use techniques of information geometry to characterise areas on the map of potentially clustered data by calculating local expansion/contraction rates in the statistical manifold of local models. Such knowledge is highly desirable for topographic map understanding, but is impossible to obtain in a principled manner from recursive extensions of SOM.

The thesis concludes with the study of a real-world problem. It formulates an extension of GTM that constructs topographic maps of light curves that originate from binary eclipsing stars. To that purpose a probabilistic physical model is formulated.

1.3 Merits of Generative Probabilistic Model-Based Formulations

As aforementioned, this thesis discusses extensions of the GTM to novel data types as well as the benefits of this approach over recursive neural-based extensions of SOM. The benefits of the approach followed here stem from its probabilistic and model-based nature.

Amongst the benefits of generative probabilistic models, is their capability of modelling uncertainty. In such models a function of likelihood arises that quantifies how well a given dataset is modelled (e.g. [Bishop, 1999, Rabiner, 1989]). The likelihood of the model is a precise objective function that allows comparisons of different model-fittings as a result of alternative choices of initialisation, training parameters and training algorithms. Furthermore, testing for overfit-

ting is possible using the robust method of cross-validation which is applicable in most practical settings, when more sophisticated methods are not available. Of course, error (cost/objective) functions and detection of overfitting are also available in other machine learning models such as discriminative models. However, one advantage of generative probabilistic models is the capability of directly comparing alternative models that may not belong to the same family.

Moreover, probabilistic approaches can handle missing data in a principled manner as for instance in [Ng et al., 2004] where the EM algorithm is employed. Also, in case overfitting is detected, measures for regularisation can be taken to improve the generalisation performance of the model. One possibility is maximum a-posteriori (MAP) estimation by the incorporation of priors on the model parameters. For example the common practice in neural networks of adding a penalty term (proportional to the norm of the weights) for regularisation purposes, is justified when a probabilistic view is assumed [Bishop, 1996]. Another feature is that such models can naturally form mixture models (e.g. mixture of probabilistic PCA models in [Bishop, 1999], e.g. mixture of hidden Markov models in [Smyth, 1997]) and also composite models such as hidden Markov models with emissions modelled as mixture of Gaussians or hierarchical models (e.g. hierarchical hidden Markov models [Fine et al., 1998]).

Furthermore, generative model based formulations inherently lend themselves to being explanatory [Smyth, 1999]; observing the underlying generative process, be it a mixture of Gaussians, a hidden Markov model or a probabilistic grammar, provides clues as to how data items arise. This may also inform us on whether the chosen model is a plausible one for a given problem.

1.4 Thesis Organisation

A recurrent theme around the three basic data types of vectors, sequences and tree-structures permeates most of the thesis:

- In chapter 2 we review SOM that processes vectorial data. SOM sets a paradigm that has inspired various extensions that introduce feedback connections to allow for the processing of data expressed as sequences and acyclic-directed graphs (trees are a special subcase of graphs). We also review probabilistic extensions of SOM. A discussion of these approaches follows.

- In chapter 3 generative probabilistic models are reviewed that model the three data types, namely the Gaussian density for vectors, hidden Markov models for sequences and hidden Markov tree models and Markov tree models for tree-structures. These generative

probabilistic models are trained via the Expectation-Maximisation algorithm that is also reviewed in the same chapter.

- In chapter 4 the GTM algorithm is reviewed that constructs topographic maps of vectorial data. We briefly mention an extension of GTM, which here we shall refer to as GTM-HMM that processes sequences [Tiňo et al., 2004], and formulate our own two extensions the GTM-HMTM and GTM-MTM that process tree-structures. Experimental results are presented and analysed. We also discuss the advantages that the generative probabilistic formulation of our approach brings compared to a candidate member of the recursive neural-based approaches.

- Chapter 5 introduces magnification factors that reveal local contractions/expansions on the topographic map. Magnification factors are derived for the GTM and the extensions presented in chapter 4. Experimental results are presented and discussed.

- In chapter 6 we depart from the processing of vectors, sequences and tree-structures and demonstrate the power of our generative probabilistic formulation by considering a real world problem. We consider light curves from eclipsing binary stars and derive a GTM[3] that constructs topographic maps of such astronomical objects. A probabilistic physical model is formulated and employed as the local noise model for the new GTM. We present the resulting topographic maps and plots of magnification factors.

- Chapter 7 concludes the thesis with a discussion of its main contributions.

1.5 Thesis Contributions and Publications

This thesis makes the following contributions:

- Develops two novel extensions of the GTM algorithm for the visualisation of tree-structured data, accompanied by a discussion comparing these extensions to a representative of recursive neural-based approaches (chapter 4).

- Develops a novel extension of the GTM algorithm for the visualisation of light curves from eclipsing binary stars (chapter 6).

- Studies two approaches for the measurement of magnification factors in topographic maps (chapter 5). The study is not limited to the extensions of GTM developed in this thesis,

[3] By 'GTM' we shall refer both to the GTM algorithm by [Bishop et al., 1998] and to the probabilistic generative model-based formulation of topographic latent models that GTM dictates.

but also concerns the original GTM and a previously developed GTM for the visualisation of sequences [Tiňo et al., 2004].

This work has lead to the following publications:

1. Peter Tiňo, Nikolaos Gianniotis: *Metric Properties of Structured Data Visualizations through Generative Probabilistic Modeling.* International Joint Conference on Artificial Intelligence 2007: 1083-1088.

2. Nikolaos Gianniotis, Peter Tiňo: *Visualisation of Tree-Structured Data through Generative Probabilistic Modelling.* European Symposium on Artificial Neural Networks 2007: 97-102.

3. Nikolaos Gianniotis, Peter Tiňo: *Visualisation of Tree-Structured Data through Generative Topographic Mapping.* Submitted to IEEE Transactions on Neural Networks: Accepted subject to minor modifications.

Chapter 2

Neural-Based Approaches to Topographic Mapping

The Self-Organising Map (SOM) [Kohonen, 1990] is the archetypal neural network algorithm for the topographic mapping of vectorial data. SOM has enjoyed considerable success in many diverse areas, a comprehensive array of applications can be found in [Kaski et al., 1998, Oja et al., 2003]. Furthermore, it has inspired numerous extensions that deal with data of non-vectorial types. An excellent overview of such extensions under a general framework can be found in [Hammer et al., 2004]. SOM and its extensions rely on a Hebbian type of learning where two processes, one of *competition* and one of *cooperation*, take place between the neurons. For the purposes of visualisation, neurons are usually organised on a two dimensional rectangular lattice. All neurons are supplied with weight vectors. At every time step a data item is presented to the network and the neuron that is closest to the pattern is declared the winning neuron. This is the competition step. The weights of the winning neuron are updated so as to increase its future activation to this particular pattern. Next at the cooperation step, the weight vectors of neighbouring neurons are also updated, albeit to a lesser extend. This type of training leads to a topographic ordering of the neurons on the lattice. Extensions of SOM to structured data types generally adhere to this framework of learning. However, in order to capture the structure of the particular data type, a notion of *context* is introduced. Structured data types, such as sequences or graphs, are processed in a recursive manner by adding feedback connections, e.g. a sequence might be processed one symbol at a time and the neural activation induced by each symbol is fed back to the network as complementary to the input of the next symbol. During such recursive processing of a data item, a context is created and recursively updated at each time step that represents the components of the data item processed until the

current competition step.

For all of the techniques presented in this section we define here some common notation. In particular each extension to SOM formulates a q-dimensional rectangular lattice of neurons indexed by $j = 1, ..., M$ (q is typically set to 2 for visualisation purposes). The location of a neuron j on the lattice is referenced by a q-dimensional vector $\boldsymbol{x}_j$. Each neuron j is supplied by a weight vector $\boldsymbol{w}_j \in \mathbb{R}^d$, where d is the dimensionality of the input space, and its activation (response) to an input $\boldsymbol{t}$ is denoted by $y_j(\boldsymbol{t})$.

2.1 Vector Quantisation

Before reviewing SOM, we consider the vector quantisation algorithm (VQ) [Gray, 1984] which may be viewed conceptually as a precursor to SOM. We consider the domain of vectors $\boldsymbol{t} \in \mathbb{R}^d$. The goal of VQ is dimensionality reduction or data compression. VQ seeks to achieve this by producing a set of M codebook vectors $\boldsymbol{w} \in \mathbb{R}^d$, that are sufficient approximations of the set of input vectors. In a practical setting, each time a vector $\boldsymbol{t}$ needs to be transmitted via a communication channel, VQ selects the closest codebook $\boldsymbol{w}$ vector, in the Euclidean distance type of sense, and transmits that instead. Provided both ends of the channel share the same codebook, only the index of the codebook needs to be transmitted which procures a gain in communication bandwidth.

VQ defines an encoding function $\gamma(\cdot)$ and a decoding function $\zeta(\cdot)$. Thus, for perfect encoding-decoding we have that $\zeta(\gamma(\boldsymbol{t})) = \boldsymbol{t}$. The distortion, that is the efficiency of the encoding-decoding process, can be measured by the mean squared error:

$$D = \int_{\boldsymbol{t}} p(\boldsymbol{t}) \left\| \boldsymbol{t} - \zeta(\gamma(\boldsymbol{t})) \right\|^2 d\boldsymbol{t}, \tag{2.1}$$

where $p(\boldsymbol{t})$ is the probability distribution of the data.

The goal is then to adjust the encoding and decoding functions so that the distortion is minimised. Training of VQ proceeds via the generalised Lloyd algorithm [Gray, 1984]:

- *Step 1.* Given input $\boldsymbol{t}$, calculate its Euclidean distance to all codebooks and return the index of the codebook with the minimum Euclidean distance. Thus, $\gamma(\boldsymbol{t}) = argmin_j \left\{ \left\| \boldsymbol{t} - \boldsymbol{w}_j \right\|^2 \right\}$.
- *Step 2.* Given codebook index $\gamma(\boldsymbol{t}) = j$, replace codebook $\boldsymbol{w}_j$ by the centroid of all vectors that γ would encode as the codebook $\boldsymbol{w}_j$. Thus, $\zeta(j) = \frac{1}{|\boldsymbol{t} \in \{\gamma(\boldsymbol{t}) = j\}|} \sum_{\boldsymbol{t} \in \{\gamma(\boldsymbol{t}) = j\}} \boldsymbol{t}$. Goto *step 1.*

The algorithm alternates between these two steps until a minimum for D is achieved.

2.2 The Original Self-Organising Map

The Self Organising Map (SOM) [Kohonen, 1990] is a neural network that can be used for the visualisation of vectorial data. We denote elements of the domain of vectorial data by $\boldsymbol{t} \in \mathbb{R}^d$. SOM may be viewed as a constrained version of VQ. The constraint is that neurons are topologically ordered so that the spatial location of a neuron bears a relationship to its weight. This means that neurons of "similar" locations on the lattice, have "similar" weights, hence represent "similar" regions of the input space. The constraint is not imposed explicitly but is a consequence of the training algorithm that introduces a lateral interaction between the neurons favouring this organisation. This contrasts with VQ where codebooks are independently updated.

Learning in SOM constitutes of two steps, a competitive step and a cooperative step. SOM alternates between the two steps for each iteration $i = 1, 2, \ldots$. Starting with the competition at each iteration i, SOM is presented with a randomly chosen input vector $\boldsymbol{t}$. The activation of each neuron j is calculated as:

$$y_j(\boldsymbol{t}) = \|\boldsymbol{t} - \boldsymbol{w}_j\|^2, \tag{2.2}$$

which is the squared Euclidean distance between the weight $\boldsymbol{w}_j$ of neuron j and the presented input $\boldsymbol{t}$. A competition is then announced amongst all neurons with the purpose of finding the best matching neuron to the presented input, i.e. the neuron whose weight has the minimum Euclidean distance to the input. This neuron is declared the winner of the competition and is denoted by $I(\boldsymbol{t})$:

$$I(\boldsymbol{t}) = \underset{j}{argmin} \left\{ y_j(\boldsymbol{t}) \right\}. \tag{2.3}$$

The cooperation step follows, which involves updating the weights of neurons so as to increase their future activation to this particular pattern. This update is conditioned by a neighbourhood defined around the winning neuron as its centre. Neurons in the neighbourhood have their weights updated depending on their distance from the centre of the neighbourhood. Thus, neurons closer to the winning neuron have their weights adjusted closer to the input pattern, than neurons that are more distant. The distance between the winning neuron $I(\boldsymbol{t})$ and a neuron j is formally defined by a neighbourhood function h of the type:

$$h(j, I(\boldsymbol{t})) = \exp(-\frac{dist(j, I(\boldsymbol{t}))}{\sigma(i)^2}), \tag{2.4}$$

where $dist(j, I(\boldsymbol{t}))$ is the lateral distance between neurons j and $I(\boldsymbol{t})$ (e.g. Manhattan distance), and $\sigma(i)$ is a parameter that controls the width of the neighbourhood. This distance is incorpo-

rated in the update rule of the weights. Moreover, a learning rate $\eta(i)$ is also included to form the following expression for updating the weights of neuron j:

$$\boldsymbol{w}_j(i+1) := \boldsymbol{w}_j(i) + \eta(i)h(j, I(\boldsymbol{t}))(\boldsymbol{t} - \boldsymbol{w}_j(i)), \tag{2.5}$$

where $\boldsymbol{w}_j(i)$ is the weight of neuron j at iteration i. The neighbourhood function possesses two important properties; it is symmetric around the winning neuron where $d(I(\boldsymbol{t}), I(\boldsymbol{t})) = 0$, and secondly it decreases monotonically. Also since $d(I(\boldsymbol{t}), I(\boldsymbol{t})) = 0$, winning neuron $I(\boldsymbol{t})$ receives the greatest update. The learning rate η and neighbourhood function h are dependent on time. These parameters adhere to a time-decaying schedule of the type:

$$\eta(i) = \eta_0 \exp(-\frac{i}{\eta_1}), \tag{2.6}$$

$$\sigma(i) = \sigma_0 \exp(-\frac{i}{\sigma_1}), \tag{2.7}$$

where the constants η_0, σ_0 define the initial values for the learning rate and neighbourhood and constants η_1, σ_1 control the rate of decay respectively. The gradual decay of η and h is essential for the topographic organisation of the neurons. Repeated presentations of input data, gradually shift the weights of the neurons which eventually approximate the probability density of the data $p(\boldsymbol{t})$.

Training of the map continues until the weights of the neurons become stable. SOM with a two (or three) dimensional lattice can be used for visualisation of the input data by projecting each input $\boldsymbol{t}$ to the q-dimensional lattice of neurons. The image of each input $\boldsymbol{t}$ in the lattice is given by the location vector $\boldsymbol{x}$ of the neuron that produces the greatest activation for input $\boldsymbol{t}$:

$$\boldsymbol{x} \leftarrow I(\boldsymbol{t}). \tag{2.8}$$

2.3 Recursive Extensions to the Self-Organising Map

In this section we review some of the representatives of the recursive extensions to SOM for the processing of structured data namely sequences and acyclic directed graphs.

The *Temporal Kohonen Map (TKM)* [Chappell and Taylor, 1993] has been designed for the processing of sequences $\boldsymbol{s} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_T]$ over $\mathbb{R}^d$. Each neuron j is equipped with a weight vector $\boldsymbol{w}_j \in \mathbb{R}^d$. At each iteration i, A single input symbol[1] $\boldsymbol{s}_t,\ t = 1, \ldots, T$ is processed in a

[1]The use of index t for symbols of sequences is in potential conflict with the notation of vectors $\boldsymbol{t}$. We choose index t because it appears more "natural" for this case of temporal data and because it widely used in the relevant

context given by the past activations of the neuron. So neurons in TKM do not lose their past activity immediately as in SOM when a new input is presented, but the context information decays gradually. When the processing of an entire string $\boldsymbol{s}$ has been completed, the past activity of all neurons is reset to zero. The activation of neuron j at iteration i for input $\boldsymbol{s}_t$, is computed as follows:

$$y_j(\boldsymbol{s}_t) = \alpha \|\boldsymbol{s}_t - \boldsymbol{w}_j(i)\|^2 + (1-\alpha) y_j(\boldsymbol{s}_{t-1}), \tag{2.9}$$

where for the activation $y_j(\boldsymbol{s}_0)$ we define that $\boldsymbol{s}_0 = \square$ is the empty sequence so that $y_j(\square) = 0$, and $\alpha \in (0,1)$ is a decay parameter. The winner for input $\boldsymbol{s}_t$ is:

$$I(\boldsymbol{s}_t) = \underset{j}{argmin} \left\{ y_j(\boldsymbol{s}_t) \right\}. \tag{2.10}$$

Training at each iteration i involves adapting weight $\boldsymbol{w}_j$ to the current input $\boldsymbol{s}_t$ in the same Hebbian fashion as in SOM, using the following rule:

$$\boldsymbol{w}_j(i+1) := \boldsymbol{w}_j(i) + \eta h(j, I(\boldsymbol{s}_t))(\boldsymbol{s}_t - \boldsymbol{w}_j). \tag{2.11}$$

The parameter η is the learning rate and $h(\cdot,\cdot)$ is a Gaussian neighbourhood function defined on pairs of neurons on the map:

$$h(j, I(\boldsymbol{s}_t)) = \exp(-\frac{dist(j, I(\boldsymbol{s}_t))}{\sigma^2}), \tag{2.12}$$

where $dist$ is the distance of neurons j and $I(\boldsymbol{s}_t)$ on the map and σ controls the neighbourhood size. Parameters η and σ are decreased with time to allow for topographic convergence as in SOM [Kohonen, 1990].

Recurrent SOM (RSOM) [Varsta et al., 1997] modifies TKM by summing the deviation of the weights $\boldsymbol{w}_j$ as opposed to distances. At iteration i the activation of neuron j for input $\boldsymbol{s}_t$ is:

$$\boldsymbol{y}_j(\boldsymbol{s}_t) = \alpha(\boldsymbol{s}_t - \boldsymbol{w}_j(i)) + (1-\alpha)\boldsymbol{y}_j(\boldsymbol{s}_{t-1}). \tag{2.13}$$

The result of this summation is a vector as opposed to a scalar in TKM and again previous inputs are utilised in a recursive manner. The winning neuron in this case is the neuron that satisfies:

$$I(\boldsymbol{s}_t) = \underset{j}{argmin} \left\{ \|\boldsymbol{y}_j(\boldsymbol{s}_t)\| \right\}. \tag{2.14}$$

Here however, adaptation takes into consideration the previous inputs, coded in $\boldsymbol{y}_j(i)$, by

literature. We also ensure that both notations to not appear in together in the same model.

adapting the weights as follows:

$$\boldsymbol{w}_j(i+1) := \boldsymbol{w}_j(i) + \eta h(j, I(\boldsymbol{s}_t))\boldsymbol{y}_j(\boldsymbol{s}_t). \tag{2.15}$$

Recursive SOM (RecSOM) [Voegtlin, 2002] takes into account the context of inputs by explicitly augmenting each unit j with a context vector $\boldsymbol{c}_j \in \mathbb{R}^q$ that represents the activations of all the units in the map at the previous iteration. The activation is computed as:

$$y_j(\boldsymbol{s}_t) = \alpha\|\boldsymbol{s}_t - \boldsymbol{w}_j\|^2 + \beta\|[exp(-y_1(\boldsymbol{s}_{t-1})), \ldots, exp(-y_M(\boldsymbol{s}_{t-1}))]^T - \boldsymbol{c}_j\|^2, \tag{2.16}$$

where α and β are positive constants that control the contribution of the weight and context vectors respectively. Training again is Hebbian for both $\boldsymbol{w}_j$ and $\boldsymbol{c}_j$:

$$\boldsymbol{w}_j(i+1) := \boldsymbol{w}_j(i) + \eta_1 h(j, I(\boldsymbol{s}_t))(\boldsymbol{s}_t - \boldsymbol{w}_j), \tag{2.17}$$

$$\boldsymbol{c}_j(i+1) := \boldsymbol{c}_j(i) + \eta_2 h(j, I(\boldsymbol{s}_t))([exp(-y_1(\boldsymbol{s}_{t-1})), \ldots, exp(-y_M(\boldsymbol{s}_{t-1}))]^T - \boldsymbol{c}_j), \tag{2.18}$$

where η_1 and η_2 are the learning rates for weight and context vector respectively. Thus, neurons do not compete only in matching their weight vector to the current input, but also their context vector to the current context of the input.

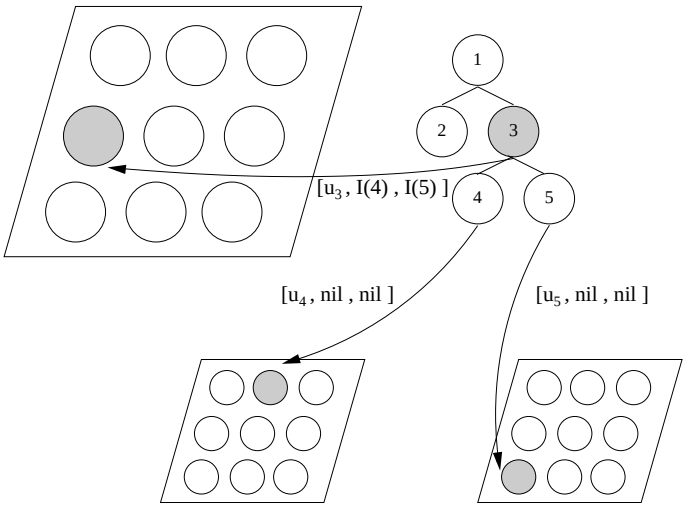


Fig. 2.1: Activation for label $\boldsymbol{u}_3$ of a tree-pattern in SOMSD: Activation is calculated bottom up, thus the children of input node 3 are processed beforehand. Since nodes 4 and 5 are leaf nodes their contexts are filled in with the special $\boldsymbol{nil}$ vector. The winner neurons $I(4)$ and $I(5)$ of the input labels $\boldsymbol{u}_4$, $\boldsymbol{u}_5$ of nodes 4 and 5 respectively are supplied as the context for node 3.

Further in the context of sequence processing, the *Merge SOM* (MSOM) is introduced in [Strickert and Hammer, 2005]. Similarly to RecSOM, each neuron is equipped with a context

vector $\boldsymbol{c}_j \in \mathbb{R}^d$. In this case, the context vector $\boldsymbol{c}_j$ does not represent the activations of the entire map, but is a combination of the weight and context $\boldsymbol{c}_{I(\boldsymbol{s}_{t-1})}$ of the previous winner. The activation of a neuron j is computed as:

$$y_j(\boldsymbol{s}_t) = \alpha\|\boldsymbol{s}_t - \boldsymbol{w}_j\|^2 + (1-\alpha)\|\boldsymbol{c}_i - \boldsymbol{c}_j\|^2, \tag{2.19}$$

where $\boldsymbol{c}_i$ is the current context and $\alpha \in [0,1]$ is a parameter that controls the contributions of current input and context. The current context vector is given by the following linear combination:

$$\boldsymbol{c}_i = \beta \boldsymbol{c}_{I(\boldsymbol{s}_{t-1})} + (1-\beta)\boldsymbol{w}_{I(\boldsymbol{s}_{t-1})}. \tag{2.20}$$

At the beginning of the processing of a sequence, the context is set equal to $\boldsymbol{c}_1 = \boldsymbol{0}$. Training again is Hebbian for both $\boldsymbol{w}_j$ and $\boldsymbol{c}_j$, applied at each time step.

SOM for Structured Data (SOMSD) presented in [Hagenbuchner et al., 2003] is an extension of SOM designed to process patterns expressed as directed acyclic graphs (trees and sequences are special cases). Each node v of a graph pattern has a label $\boldsymbol{u}_v \in \mathbb{R}^d$. Each neuron j besides its weight vector $\boldsymbol{w}_j \in \mathbb{R}^d$ is supplied with k additional coordinate vectors $\boldsymbol{c}_j \in \mathbb{R}^q$, where k is the maximum out-degree of the graphs in the dataset. Similarly to RecSOM, these additional vectors try to capture the context of the current input. The context is provided by the winning neurons $I(j)$ of children $j = 1, 2, .., k$ of the node v currently processed. Thus, each context vector $\boldsymbol{c}_j$ tries to match winning neuron $I(j)$. The augmented input to SOMSD is:

$$[\boldsymbol{u}_v, I(1), I(2), \ldots, I(k)]\,.$$

The activation of neuron j is calculated as:

$$y_j(v) = \mu_1\|\boldsymbol{u}_v - \boldsymbol{w}_j\|^2 + \mu_2(\|I(1) - \boldsymbol{c}_1\|^2 + \cdots + \|I(k) - \boldsymbol{c}_k\|^2), \tag{2.21}$$

where μ_1 and μ_2 are positive constants that control the contribution of the input label $\boldsymbol{u}_v$ and the context vectors $\boldsymbol{c}_i$. Processing of input items proceeds in a bottom-up fashion: before a node v can be processed all of its children must be already processed. This is illustrated in Fig. 2.1. Therefore, processing starts from the leaf nodes (nodes without children). When a leaf node is processed the context vectors are set to some default values representing the empty tree $\boldsymbol{nil}$. The same applies for nodes with less than k children where the coordinate vectors $\boldsymbol{c}_j$ of the missing children are substituted by $\boldsymbol{nil}$. The coordinate vector of $\boldsymbol{nil}$ is chosen to be $(-1, \ldots, -1)$ so that it resides outside the lattice. SOMSD is trained in a Hebbian fashion and as is usual in

SOM-type formulations, the learning rate and the neighbourhood radius decay gradually. The winner is the neuron with the closest weight and context vectors to the augmented input:

$$I(v) = \underset{j}{argmin} \left\{ y_j(v) \right\}. \tag{2.22}$$

If μ_1 is set to 1 and μ_2 is set to 0, SOMSD reduces to the standard SOM algorithm.

2.4 Probabilistic and Kernel Extensions of the Self-Organising Map

The approaches presented in the previous section do not use an explicit model for the data and rely on the neural network to develop an internal suitable representation.

In [Hollmén et al., 1999] a *Self-Organising Map for Clustering Probabilistic Models* is presented, where each neuron stores as its weight a vector $\boldsymbol{w}_j$ that contains the parameters of a probabilistic model. Thus, each neuron parametrises a local density $p(\cdot|\boldsymbol{w}_j)$. The setting of this study is the exploration of temporal data and more specifically of user-profiles on a mobile communications network. User profiles are binary sequences $\boldsymbol{s} = \{s_1, \ldots s_T\}, s_t \in \{0, 1\}$ of length T. A call is described as an observed process of transition cases: $(s_t = 0, s_{t-1} = 1)$ is the beginning of a call, $(s_t = 1, s_{t-1} = 0)$ is the end of a call, $(s_t = 1, s_{t-1} = 1)$ is an on-going call and $(s_t = 0, s_{t-1} = 0)$ is on-going silence. The probabilistic model for user-profiles is a process governed by a transition matrix $\boldsymbol{B}$ with entries of probabilities $b_{kl} = p(s_t = l|s_{t-1} = k)$. The sum of entries of matrix $\boldsymbol{B}$ of each row must equal to 1. The model instantiated by neuron j is:

$$\begin{aligned} p(\boldsymbol{s}|\boldsymbol{w}_j) &= \prod_{t=1}^{T} p(s_t = l|s_{t-1} = k, \boldsymbol{w}_j), \\ \log p(\boldsymbol{s}|\boldsymbol{w}_j) &= \sum_{t=1}^{T} \log p(s_t = l|s_{t-1} = k, \boldsymbol{w}_j). \end{aligned} \tag{2.23}$$

The log-likelihood in (2.23) is used to determine winner $I(\boldsymbol{s})$ for input $\boldsymbol{s}$ presented at the competition step:

$$I(\boldsymbol{s}) = \underset{j}{argmax} \left\{ \sum_{t=1}^{T} \log p(s_t = l|s_{t-1} = k, \boldsymbol{w}_j) \right\}, \tag{2.24}$$

The log-likelihood in (2.23) is also used in the adaptation rule:

$$\boldsymbol{w}_j(i+1) := \boldsymbol{w}_j(i) + \eta h(j, I(\boldsymbol{s})) \frac{d}{d\boldsymbol{w}_j} \log p(\boldsymbol{s}|\boldsymbol{w}_j(i)) \tag{2.25}$$

where neighbourhood function h and the learning rate η are reused from (2.4) and (2.6) respectively. Note that this update rule can lead to invalid parameters (parameter are probabilities which must be restricted in $[0, 1]$). Thus, measures in the form of a suitable parametrisation or constraints must be taken. In [Hollmén et al., 1999] the solution of introducing a suitable parametrisation is used.

In [Kaski et al., 2001] a SOM formulation is presented in the setting of bankruptcy analysis, where the winner neuron is determined by a distance metric based on the Fisher information matrix. Even though this approach is concerned with vectorial data, it is of interest since its aim is to use a data-driven metric instead of the customary Euclidean distance metric. The data are financial statements of companies and are pairs of feature vectors $\boldsymbol{t} \in \mathbb{R}^d$, describing certain indicators such as growth, profitability etc. of a company, and binary indicator variables $c \in \{0, 1\}$ signifying the bankruptcy risk of the company in the next three years. The goal is to achieve a topographic organisation of the data $\boldsymbol{t}$ driven by the implicit information present in c that indicates which features are relevant to the task of bankruptcy risk analysis. A prediction of bankruptcy risk for a statement $\boldsymbol{t}$, is expressed as a conditional density $p(c|\boldsymbol{t})$ which is (as in many other application domains) unknown. The joint density $p(c, \boldsymbol{t})$ is learnt from the data, and the conditional density $p(c|\boldsymbol{t})$ is then obtained via Bayes' rule.

Small displacements in the conditional density, i.e. $p(c|\boldsymbol{t}+\boldsymbol{dt})$, reveal how variable c changes depending on the directions $\boldsymbol{dt}$. Such small changes can be measured by the KLD, which locally is:

$$D_{KL}[p(c|\boldsymbol{t})||p(c|\boldsymbol{t}+\boldsymbol{dt})] = \boldsymbol{dt}^T \boldsymbol{F}(\boldsymbol{t}) \boldsymbol{dt}, \tag{2.26}$$

where $\boldsymbol{F}(\boldsymbol{t})$ is the Fisher information matrix evaluated at $\boldsymbol{t}$, that reveals local scaling factors. On the SOM lattice, each neuron j is equipped with a weight vector $\boldsymbol{w}_j \in \mathbb{R}^d$. The weight vector parametrises a local conditional density $p(c|\boldsymbol{w}_j)$. When SOM is presented with input $\boldsymbol{t}$ at competition step, winner $I(\boldsymbol{t})$ is determined by:

$$\begin{aligned} I(\boldsymbol{t}) &= \underset{j}{argmin} \left\{ D_{KL}[p(c|\boldsymbol{w}_j)||p(c|\boldsymbol{t})] \right\} \\ &= \underset{j}{argmin} \left\{ \boldsymbol{dt}^T \boldsymbol{F}(\boldsymbol{w}_j) \boldsymbol{dt} \right\}, \end{aligned} \tag{2.27}$$

where $\boldsymbol{dt}$ is the Euclidean distance between $\boldsymbol{w}_j$ and $\boldsymbol{t}$. The metric in (2.26) should strictly be used for computing local distances within the neighbourhood of a neuron j, while non-local distances should be calculated as path integrals. However, the same metric is also used to approximate non-local distances by assuming that it will be locally accurate and that neighbours with a

greater separating distance $\boldsymbol{dt}$ will still be calculated as being farther than close neighbours. Thus, the winner should still be determined accurately. Although the winner is determined by metric (2.26), the update rule for the weights is the same as in the original SOM.

An interesting approach is undertaken in [Verbeek et al., 2005] where *Self-organising mixture models* (SOMM) are developed. The core idea of the approach is that a topology can be introduced into a mixture model, by modifying the E-step of the EM algorithm. The approach can be extended to any probabilistic model that can be optimised via EM, hence various data types can be accommodated. Here we consider vectors $\boldsymbol{t}$ and a mixture model of C Gaussian components $p(\boldsymbol{t}^{(n)}|\boldsymbol{\theta}_c)$:

$$p(\boldsymbol{t}|\boldsymbol{\Theta}) = \sum_{c=1}^{C} p(c)p(\boldsymbol{t}^{(n)}|\boldsymbol{\theta}_j), \tag{2.28}$$

where $\boldsymbol{\Theta}$ encapsulates vectors $\boldsymbol{\theta}_j$, the parameters of the individual Gaussian components. Each component j is associated with a coordinate vector $\boldsymbol{x}_j$ which for visualisation purposes is two-dimensional. Coordinate vector $\boldsymbol{x}_j$ determines the position of the component in a latent space that will be used to project the high-dimensional dataset. To that purpose coordinate system $\boldsymbol{x}$ is defined as a rectangular grid in the latent space.

When using EM to train a mixture of Gaussians, in the E-step the posterior distribution of the components is estimated. The posterior may be factorised for independently generated data, and we denote the posterior distribution concerning the n-th data item by $p(j|\boldsymbol{t}^{(n)})$. The posterior is used in the M-step to weigh the contribution of each data item $\boldsymbol{t}^{(n)}$ in updating the parameters of each component j (for more details see 3.2.1). Intuitively, posterior $p(j|\boldsymbol{t}^{(n)})$ expresses the responsibility that component j bears for generating data item $\boldsymbol{t}^{(n)}$. In SOMM, the E-step is modified in order to introduce a sense of topology. Viewing posteriors $p(j|\boldsymbol{t}^{(n)})$ as functions of components j, we restrict them to smooth, normalised distributions of the form:

$$h_k(j) \propto \exp(-\sigma\|\boldsymbol{x}_k - \boldsymbol{x}_j\|^2), \text{ with } \sum_{j=1}^{M} h_k(j) = 1, \tag{2.29}$$

where $k = 1, \ldots, C$ is an index over components and σ controls the radius of the distributions. Each distribution h_k is centred at a component k and acts as a neighbourhood function similarly to the one defined in SOM. In the modified E-step, for each input $\boldsymbol{t}^{(n)}$ we choose a neighbourhood function h_k that minimises $D_{KL}[h_k||p(j|\boldsymbol{t}^{(n)})]$ as the posterior probability given $\boldsymbol{t}^{(n)}$. This introduces a lateral interaction between the mixture components when it comes to updating the parameters in the M-step; for $p(j|\boldsymbol{t}^{(n)}) = h_k$ component k receives the most responsibility, while neighbouring components also receive responsibility, albeit to a lesser degree.

In the setting of Gaussian mixtures, the E- and M-step translate to the following update equations:

1. E-step: $p(j|\boldsymbol{t}^{(n)}) = argmin_{h_k} \left\{ D_{KL}[h_k||p(j|\boldsymbol{t}^{(n)})] \right\}$
2. M-step: Perform standard M-step, using the posteriors from modified E-step.

Parameter σ may be annealed from a wide neighbourhood function to a small neighbourhood with an appropriate schedule. Once training has been completed, each data point $\boldsymbol{t}^{(n)}$ is mapped to the latent space as the mean of the posterior distribution over the latent space:

$$proj(\boldsymbol{t}^{(n)}) = \sum_{j=1}^{C} p(j|\boldsymbol{t}^{(n)})\boldsymbol{x}_j. \tag{2.30}$$

In [Hofmann, 2000] *ProbMap* is introduced, which constitutes a probabilistic approach to visualising document collections. ProbMap learns K (K is a parameter of the algorithm) thematic topics from a document collection and organises them on a two-dimensional lattice so that neighbouring topics are similar. Each neuron indexed by $k = 1, \ldots, K$ on the SOM lattice represents one of the K topics to be learnt. A dataset of documents is a set $\mathcal{D} = \{d^{(1)}, ..., d^{(N)}\}$, where $d^{(n)}$ with $1 \leq n \leq N$ is a document. Each document consists of $T(n)$ number of words from a fixed vocabulary $\boldsymbol{V} = \{w_1, ..., w_R\}$, thus $d^{(n)} = (d_1^{(n)}, \ldots\ d_{T(n)}^{(n)})$ with $d_t^{(n)} \in \boldsymbol{V}$. Each document $d^{(n)}$ has a parameter vector of probabilities $\tau^n = [p(1|d^{(n)}) \ldots p(K|d^{(n)})]$, where $p(k|d^{(n)})$ quantifies the degree to which document $d^{(n)}$ expresses thematic topic k, with $\sum_{k=1}^{K} p(k|d^{(n)}) = 1$. Also, for each topic k a parameter vector of probabilities $\phi^k = [p(w_1|1) \ldots p(w_R|K)]$ is defined where $p(w_r|k)$ expresses the probability that word w_r occurs under topic k, with $\sum_{r=1}^{R} p(w_r|k) = 1$. A fixed, symmetric neighbourhood function is imposed on the SOM lattice that returns the distance between two neurons k and l:

$$p(l|k) = \frac{\exp(-\sigma\ h(k,l)^2)}{\sum_{l'=1}^{K} \exp(-\sigma\ h(k,l')^2)}, \tag{2.31}$$

where σ controls the radius of the neighbourhood and $h(k,l)$ is the Euclidean distance between neurons k and l on the lattice. Thus, a distance is imposed on the K topics to be learnt.

ProbMap is a generative model operating as follows. A document $d^{(n)}$ is produced by generating one word $d_t^{(n)}$ at a time. For each $t = 1, \ldots, T(n)$ a random topic k is chosen with probability $p(k|d^{(n)})$. Chosen topic k is corrupted with a probability $p(l|k)$ into a second topic l. However, the nature of $p(l|k)$ is such that a topic is more likely to be corrupted into a topic represented by neurons close to neuron k than into a topic represented by neurons distant to k.

This introduces the topographic organisation of the topics. Having chosen the second topic l, a word for $d_t^{(n)} = w_r$ is generated with probability $p(w_r|l)$.

The model likelihood given dataset $\mathcal{D}$ reads:

$$\mathcal{L}(\tau, \phi|\mathcal{D}) = \prod_{n=1}^{N} \sum_{k_1^{(n)}=1}^{K} \sum_{l_1^{(n)}=1}^{K} \cdots \sum_{k_{T(n)}^{(n)}=1}^{K} \sum_{l_{T(n)}^{(n)}=1}^{K} \prod_{t=1}^{T(n)} p(k_t^{(n)}|d^{(n)}) p(l_t^{(n)}|k_t^{(n)}) p(d_t^{(n)}|l_t^{(n)}), \quad (2.32)$$

where vectors $k_t^{(n)}$ and $l_t^{(n)}$ refer to the topics responsible for generating the word $d_t^{(n)}$. The model likelihood is maximised via the EM algorithm by postulating a set of hidden variables that indicate which states were responsible for the generation of each word. Adjusting parameters τ and ϕ in this framework leads to an organisation of K thematic topics on the two-dimensional lattice.

Departing from the probabilistic extensions of SOM, in [András, 2002] SOM is modified to incorporate a kernel function in calculating activations of neurons. A kernel function is a function K that calculates the inner product $K(\boldsymbol{t}, \boldsymbol{t}') = F(\boldsymbol{t})^T F(\boldsymbol{t}')$ of data items $\boldsymbol{t} \in \mathbb{R}^d$ embedded in a high dimensional space $\mathbb{R}^{d'}$ $(d' > d)$, via a function $F : \mathbb{R}^d \to \mathbb{R}^{d'}$ that transforms the originally linearly inseparable dataset into a linearly seperable dataset. Although, topology preservation is important here, the introduction of a kernel function aims to improve the classification ability of SOM: a trained SOM may be used for classification by assigning each data item $\boldsymbol{t}$ the class of the closest neuron j, i.e. $class(\boldsymbol{t}) = class(argmin_j \left\{ y_j(\boldsymbol{t}) \right\})$.

As stated in [András, 2002], SOM learns a Voronoi tessellation of the data space, which in the case of a linearly inseparable dataset is a difficult task that requires a large lattice (i.e. high number of neurons) in order to learn an accurate topographic map. However, if the data are linearly seperable the task is easier and a smaller lattice suffices. The activation of a neuron j given transformed data item $F(\boldsymbol{t})$ is $y_j(\boldsymbol{t}) = \|F(\boldsymbol{w}_j) - F(\boldsymbol{t})\|^2$, where weights $\boldsymbol{w}$ are also embedded into $\mathbb{R}^{d'}$. We may rewrite F in terms of K:

$$\begin{aligned} y_j(\boldsymbol{t}) = \|F(\boldsymbol{w}_j) - F(\boldsymbol{t})\|^2 &= (F(\boldsymbol{w}_j) - F(\boldsymbol{t}))^T (F(\boldsymbol{w}_j) - F(\boldsymbol{t})) \\ &= F(\boldsymbol{t})^T F(\boldsymbol{t}) + F(\boldsymbol{w}_j)^T F(\boldsymbol{w}_j) - 2F(\boldsymbol{t})^T F(\boldsymbol{w}_j) \\ &= K(\boldsymbol{t}, \boldsymbol{t}) + K(\boldsymbol{w}_j, \boldsymbol{w}_j) - 2K(\boldsymbol{t}, \boldsymbol{w}_j). \end{aligned} \quad (2.33)$$

Hence, the winner $I(\boldsymbol{t})$ is determined by:

$$I(\boldsymbol{t}) = \underset{j}{argmin} \left\{ K(\boldsymbol{t}, \boldsymbol{t}) + K(\boldsymbol{w}_j, \boldsymbol{w}_j) - 2K(\boldsymbol{t}, \boldsymbol{w}_j) \right\}. \quad (2.34)$$

Also the update rule is formulated as:

$$\begin{aligned} \boldsymbol{w}_j(i+1) &:= \boldsymbol{w}_j(i) + \eta h(j, I(\boldsymbol{t})) \frac{\partial}{\partial \boldsymbol{w}} y_j(\boldsymbol{t}) \\ &:= \boldsymbol{w}_j(i) + \eta h(j, I(\boldsymbol{t}))(\frac{\partial}{\partial \boldsymbol{w}} K(\boldsymbol{w}_j, \boldsymbol{w}_j) - \frac{\partial}{\partial \boldsymbol{w}} 2K(\boldsymbol{t}, \boldsymbol{w}_j)). \end{aligned} \tag{2.35}$$

Of course one must choose the form of K. In [András, 2002] the kernel function $K(\boldsymbol{t}, \boldsymbol{t}') = K(\boldsymbol{t}, \boldsymbol{t}'; \boldsymbol{\omega})$ is parametrised by parameter vector $\boldsymbol{\omega}$ which allows the adaptation of the kernel. The learning of the kernel function is supervised and is performed via the LVQ algorithm. Once SOM has been trained, the neurons are labeled e.g. by the majority class of data items in Voronoi cell of the neuron. Parameters $\boldsymbol{\omega}$ are set so that the best classification labels for the neurons are found. Adapting parameter $\boldsymbol{\omega}$ changes the the boundaries of the Voronoi cell and consequently the membership of data points to the Voronoi cells in directions that better classifications may be obtained.

In [Günter and Bunke, 2002] SOM is extended to work with data expressed as graphs. The neuron associated weights $\boldsymbol{w}$ are no longer feature vectors (codebooks) but explicit graph representations. Since graph data are not expressed in the convenient form of vectors, one must define a suitable distance function that measures affinities between data vectors and the representations stored in the neurons of the lattice, and also formulate an update rule that decreases the distances between data vectors and representations in neurons. The first requirement is satisfied by employing the graph edit distance d_{edit} which is defined as the least number of operations that are necessary to transform a graph $\boldsymbol{g}$ into a graph $\boldsymbol{g}'$. Edit operations include the insertion, deletion or relabelling of a node or edge. Thus, the less operations needed to transform $\boldsymbol{g}$ into $\boldsymbol{g}'$, the less the distance $d_{edit}(\boldsymbol{g}, \boldsymbol{g}')$ between them. Moreover, edit operations are associated with non-negative cost values. This makes the graph edit distance more robust to distortions (noise) present in graph data (e.g missing edges or nodes) by typically assigning low costs to the edit functions that remedy the most frequent type of distortions. The winner neuron is defined as:

$$I(\boldsymbol{g}) = \underset{j}{argmin} \left\{ d_{edit}(\boldsymbol{g}, \boldsymbol{w}_j) \right\}. \tag{2.36}$$

By applying only a subset of the necessary edit operations that transform completely graph $\boldsymbol{g}$ so that $d_{edit}(\boldsymbol{g}, \boldsymbol{g}') = 0$, we obtain a graph $\boldsymbol{g}^o$ for which $d_{edit}(\boldsymbol{g}, \boldsymbol{g}') = d_{edit}(\boldsymbol{g}, \boldsymbol{g}^o) + d_{edit}(\boldsymbol{g}^o, \boldsymbol{g}')$. A function f is defined that for a given distance a with $0 \leq a \leq d_{edit}(\boldsymbol{g}, \boldsymbol{g}')$, it returns a graph $f(a, \boldsymbol{g}) = \boldsymbol{g}^o$ such that the distance $d_{edit}(\boldsymbol{g}, \boldsymbol{g}^o)$ approximates a as closely as possible. This is useful for the update rule that needs to update the neurons in varying degrees for a given input

data item. The update for a neuron j is:

$$\boldsymbol{w}_j := f(a, \boldsymbol{g}), \tag{2.37}$$

where a must relate to a decreasing learning rate η and the distance on the lattice $h(j, I(\boldsymbol{g}))$. Neurons may be initialised by assigning to $\boldsymbol{w}$ a perturbed subset of the training data.

2.5 Discussion of Reviewed Extensions

Although, the recursive extensions of SOM have shown good results in various experimental settings, they are challenged by certain theoretical problems. Neighbourhood preservation is an important property of a good topographic map. It refers to the property of data points maintaining their neighbours after projecting them into the low-dimensional visualisation space. A map may sustain defects such as violations in neighbourhood relations or abrupt discontinuities when a neuron appears to be significantly different to its neighbouring neurons. Of course alteration of certain neighbourhood relations is inevitable since the reduction in dimension results to a loss of certain topographic information. SOM and its recursive extensions rely on Hebbian learning that does not explicitly optimise a certain neighbourhood preservation criterion. Thus, apart from visually inspecting a map, judging objectively the quality of the achieved topographic organisation is not straightforward.

Numerous suggestions of such quality measures have been put forward in the related literature. For example, quantisation error [Pölzlbauer, 2004] is a measure of how well the input patterns match their winning neurons (in a Euclidean sense of distance). However, such a measure does not respect topological aspects of the map. Another is topographic error [Pölzlbauer, 2004], that does take into consideration the topology, but in a restricted way. Topographic error tests whether for an input item the first and second best matching neurons are adjacent on the SOM lattice. If they are, then the mapping is locally continuous, otherwise a discontinuity occurs. Another example of such measures is the topographic product [Bauer and Pawelzik, 1992]. The method measures distances of neurons on the lattice of the map and in the data space (weight space). The intuition is that neurons with similar weights are neighbours in the data (weight) space and for a topographically organised map such neurons should also be neighbours on the lattice. The topographic product is concerned with preserving the relative ordering of neighbours, instead of the absolute distances that separate them. Violations in the relative ordering signify topographic defects. Furthermore in [Venna and Kaski, 2001], two measures are proposed for two types of error: neighbourhood preservation addresses the issue of a data point maintaining

its neighbours in the data space also after projection on the lattice, and trustworthiness refers to the error made when data points that are mapped close to each other on the map are in fact quite dissimilar.

Such criteria define a certain basis for evaluating topographic preservation of maps obtained by SOM and its extensions (for static data). Even though they do capture certain desirable properties of topographic preservation, we note that they are not related to some kind of objective that guides the optimisation of SOM. Therefore it is inappropriate to use these criteria in evaluating a trained map, since the map is not trained toward yielding favourable values according to the criteria. One can of course evaluate the topographic map at each iteration during training according to some criterion of topographic preservation and stop the training if no further progress (or degradation of the criterion) is observed. However, still in this case, it is not clear that SOM's Hebbian-type of training should monotonically increase the quality criterion at hand; maybe certain steps that appear counter-productive according to the criterion are necessary in order for the map to fit the data.

The problems above are related to the fact that SOM and its extensions do not formulate an objective function that quantifies the level of topographic ordering of the concerned maps, that could be used in training the maps. The SOM algorithm [Kohonen, 1990] does not explicitly state what is being optimised. Strictly speaking, it is proved in [Erwin et al., 1992] that the update rule of SOM does not constitute a gradient of any objective function. Nevertheless, in [Heskes, 1999] a slight variant of SOM is introduced that does possess such an objective function. The modification simply redefines the winning neuron as:

$$I(\boldsymbol{t}) = \underset{j}{argmin} \sum_{k=1}^{M} h(j,k) \|\boldsymbol{t} - \boldsymbol{w}_k\|. \tag{2.38}$$

Hence, the winner in this variant is not simply the neuron closest to the current input, but the neuron that minimises the quantisation error in the local neighbourhood. The form of the update rule remains the same, taking into account that the winner is given by (2.38). This idea of the winner neuron minimising quantisation error in the local neighbourhood, was used in [Hammer et al., 2004] to derive a cost function for recursive neural-based approaches. Computing such a cost function involves computing the activations of the neurons for the given input. Optimising the cost function can then be used to update the weights of the network. However, updating the weights so that the future activations of neurons match the input closer, has an additional effect; changing the activations, changes the cost function. Thus, a cycle is introduced: the cost function depends on the activations of the neurons, and the neurons are

updated according to the cost function in order to achieve better activations for the inputs. Thus, we are not faced with an ordinary optimisation problem, but something more complex that is difficult to interpret. Note, that this is not the case in SOM: in SOM an immutable Euclidean metric is defined to calculate neuron activations. Therefore, the cost function in [Heskes, 1999] does not suffer from this problem.

Another limitation of the SOM paradigm, mentioned in [Bishop et al., 1996], is that no density model is provided for the data. This deems the treatment of new incoming data items, after training has been completed, problematic. In SOM the location of a new item may be predictable, since we expect it to be placed closed to neurons of weights similar to the input in the Euclidean sense. In recursive neural-based approaches, discussed in section 2.3, the neural dynamics that dictate the placement of a new point are not as easily understood. In these approaches the input is augmented by a context, the formation of which is not easily interpreted. In SOM it is understood what it means to compare an input data item to the weights of neurons and why updating neurons of high response is beneficial. In recursive approaches two comparisons take place when presenting an input to the network. Again, as in SOM, the actual input is compared to the weights of neurons, but also the context of the augmented input is compared with the context of the augmented weights. The significance of comparing the contexts and updating them in a Hebbian way, is not easily understood as for example in the case of RecSOM in (2.16) and (2.18).

In connection to the difficulty of understanding projections, recursive neural-based approaches do not to define (at least not explicitly) a clear notion of data similarity. As aforementioned, they rely on the training process to form appropriate internal representations that capture data similarities, and this in an unsupervised setting. Thus, recursive neural-based approaches are faced with solving two tasks at the same time without external guidance: evolve an appropriate notion of similarity while striving for topographic organisation. For example, SOMSD is capable of processing graph-structured data as well as trees and sequences. These three data types can be viewed as special subcases of each other, i.e. sequences are trees with outdegree equal to one, and trees are directed-acyclic graphs where each node has a single parent. However, if we were to construct a topographic map via SOMSD on a mixed dataset of graphs, trees and sequences, it would be difficult to understand the learnt distance metric. The absence of a notion of similarity makes it difficult to understand what the distances between projected data points on topographic maps constructed by recursive approaches mean. It is not clear whether distances between the projected points on the map can be measured according to some criterion, and whether this criterion constitutes a proper metric that satisfies the

triangle inequality. Furthermore, depending on the nature of the problem, various similarity notions may be appropriate, even within a single data type. Interpretation of data is of course problem dependent and datasets of the same data structure may require a different similarity interpretation.

This brings us to the next point, that recursive neural-based approaches do not allow any expressive control over the notion of data similarity to the user. This is important because data similarity drives the organisation of the map. SOMSD does provide some control via parameters μ_1 and μ_2 regarding the contribution of labels and context. However, if hypothetically we were interested in sequences whose middle part (or alternatively their prefix or suffix) we knew was more important than the rest of the sequence in judging similarity, the necessary modifications to impose this are not obvious. It seems that such requirements cannot be easily accommodated.

The probabilistic extensions to SOM mentioned in section 2.4 seem to alleviate certain problems. In the Self-Organising Map for Clustering Probabilistic Models [Hollmén et al., 1999], a similarity metric is clearly defined in (2.23) that is also used in determining the winning neuron. Neurons in this approach instantiate probabilistic models, the generative nature of which clearly explains why a data item is assigned to its location. However, the update rule in (2.25), fashioned after the update rule in SOM (2.5), seems to be problematic to a certain extent. The goal of the update rule, as interpreted in SOM, is of course to update all neurons in the neighbourhood so that the likelihood-response of each neuron to the current input is increased in the future. After such an update, the winner neuron should remain the winner neuron for the current input since it receives the greatest update after all. Although this is indeed the case in SOM, it is not necessarily in this setting for two reasons. Firstly, in this extension we update the weight of a neuron $\boldsymbol{w}_j$, that parametrises a probabilistic model, with a gradient rule. Thus, if the step size taken in the update rule in the direction of the gradient is inappropriate, it could happen that the likelihood of a neuron would decrease instead of increase. Also, this could result in the likelihood of the winner neuron becoming less than the likelihood of one of its neighbouring neurons regarding the current input. Secondly and more importantly, the weights of neighbouring neurons may be quite different to each other and thus the parametrisation of their induced probabilistic models may differ (due to random initialisation perhaps). This means that the gradients in the update rule, are optimising different regions of the parameter space of the probabilistic model. The current learning rate and value of the neighbourhood function may then have a considerable effect on the adaptation of the likelihoods. Neighbouring neurons with different parametrisation may not express statistically similar density models. On these two accounts, the update rule in (2.25) does not appear to enforce a strict topological ordering

on the lattice of neurons or at least not in the sense that SOM does.

In [Kaski et al., 2001], where a SOM formulation is presented in the setting of bankruptcy analysis, a data driven metric is defined in the form of the conditional $p(c|\boldsymbol{t})$, which implicitly describes similarity relationships. The improvement of this approach is the decoupling of the two tasks, that of learning a similarity metric and that of producing a topographic mapping. On the contrary the recursive neural-based approaches, seem to treat the two as a single monolithic task. Training in [Kaski et al., 2001], however, does not rely on an explicit objective function which makes it problematic as aforementioned, rendering the comparison of different map formations challenging.

SOMM [Verbeek et al., 2005] is a model based on a sound probabilistic formulation. The objective function is the log-likelihood of the model and is maximised via the EM algorithm. The topology is enforced by a fixed regularisation that ensures that neighbouring components are similar. Another advantage of SOMM is its capability to handle missing values in a principled way. Setting of parameter σ of the neighbourhood functions in (2.29) seems to be a problem, as it is not part of the learning process. The framework is readily extended to other models that can be optimised via EM. Similarly, ProbMap [Hofmann, 2000] is another probabilistically sound model that possesses a clearly defined objective function, maximised also via the EM algorithm. Again setting parameter σ of the neighbourhood functions in (2.31) appears problematic as it is not included in the learning process. Furthermore in both SOMM and ProbMap the neighbourhood functions play the role of the responsibilities (posterior probabilities) of the EM algorithm. However, the fact that they are predefined and symmetrical, restricts the degree to which they can approximate the true responsibilities. For example when an input is presented, two neurons that are equidistant from the winner should not receive the same updates, unless they express the exact same responsibility (posterior) for the input. Nevertheless, because the neighbourhood functions are predefined and symmetrical, equidistant neurons receive the same update even though their true responsibilities for the input may be different.

The approaches presented in [András, 2002] and [Günter and Bunke, 2002] are also characterised by the lack of an objective cost function. Both approaches replace the standard Euclidean metric employed by SOM with alternatives that may appear more suitable and are application dependent: in [András, 2002] the kernel function operates in a high dimensional space where data can be linearly separated, and in [Günter and Bunke, 2002] the specialised distance function measures the distance between graphs as the length of a sequence of edit operations. In [András, 2002] the kernel function is adapted after the training of SOM which means that SOM may work with a transformed dataset where the inter-distances of clusters are distorted. This

can happen because of the non-linearity of the kernel, which may distort distances between data points and place them farther apart or closer in the high dimensional space (although local neighbourhoods will be preserved due to the continuity of the kernel). This can affect the visualisation quality of the topographic map as clusters may not be placed next to their "kindred" clusters. In [Günter and Bunke, 2002] the noise present in the graph data is handled by associating edit operations with costs. No mechanism is provided to learn these costs from the dataset, which are domain dependent and must be set by the user experimentally. This is important because a data driven mechanism could adapt to the dataset and provide a higher quality visualisation result by deciding which edit operations are significant in deciding the similarity of two graphs for the particular dataset.

The probabilistic extensions of SOM mentioned here share a common element, that of making use of explicit similarity measures. On the contrary the recursive neural-based extensions do not define a clear notion of similarity. We view that a visualisation problem constitutes of two components; a suitable notion of similarity or distance between the data items and the parametrisation of a visualisation lattice. A step toward this direction is taken in this work by defining the notion of data similarity via generative probabilistic models that bear a certain plausibility of generating the data at hand. A topographic organisation is introduced in these generative models via a suitable parametrisation that constrains the models on a low-dimensional space. Models such as the hidden Markov model, define a generative process for the data items they produce/model and can be used to measure similarities for data items (e.g. [Falkhausen et al., 1995]). The next chapter reviews a collection of generative probabilistic models that are employed in the subsequent chapter as components of a constraint mixture used for constructing topographic maps.

Chapter 3

Density Modelling

The intention of this chapter is to discuss some fundamental concepts necessary for developing probabilistic topographic mapping algorithms later in chapter 4. Density modelling is a prerequisite, as it is necessary to obtain a probabilistic representation of the data. Here we shall present four generative models for density modelling, namely the Gaussian distribution for vectors, hidden Markov Models for sequences, hidden Markov tree models and Markov tree models for tree structures. A single model of these types is capable of sufficiently approximating a single-class density. More complex densities can be accommodated by mixture models that employ the aforementioned models as components. A mixture model is a more flexible structure, able to accommodate complex datasets by devoting subsets of its components to modelling different partitions of the dataset. Mixture models are particularly amenable to the Expectation-Maximisation (EM) algorithm. EM constitutes an elegant procedure that iteratively estimates the parameters of a mixture by introducing a distribution of unobserved variables, each of them expressing how much responsibility each mixture component bears in explaining each data item. Thus, we shall also briefly review key points of the EM algorithm. The chapter culminates with a discussion of how constraints can be introduced to the parameters of a mixture model to convert it into a constrained mixture model, the precursor of developing algorithms for topographic organisation.

3.1 Modelling Vectorial Data

3.1.1 Unimodal Density Modelling

We consider the domain of vectorial data with $\boldsymbol{t} \in \mathbb{R}^d$. A dataset of independently generated data is $\mathcal{D} = \left\{\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \ldots, \boldsymbol{t}^{(N)}\right\}$. We choose to model $\mathcal{D}$ with a Gaussian density [Bishop, 1999]:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{t}^{(n)}|\boldsymbol{\theta}), \tag{3.1}$$

where $\boldsymbol{\theta}$ is a vector that refers to the parameters of the Gaussian density, namely the mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$. The form of the density is:

$$p(\boldsymbol{t}|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{t}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} \det(\boldsymbol{\Sigma})} \exp(-\frac{1}{2}(\boldsymbol{t} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{t} - \boldsymbol{\mu})). \tag{3.2}$$

The likelihood $\mathcal{L}(\boldsymbol{\theta}|\mathcal{D})$ of the Gaussian density is a function of parameters $\boldsymbol{\theta}$ of the model that expresses the occurrence of the sample $\mathcal{D}$ at hand:

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3.3}$$

One usually works with the log-likelihood instead:

$$\log \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \sum_{n=1}^{N} \log \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3.4}$$

Taking the derivatives of $\log \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})$ with respect to the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, we obtain the following update equations [Bishop, 1999]:

$$\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{t}^{(n)}, \tag{3.5}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{t}^{(n)} - \hat{\boldsymbol{\mu}})^T (\boldsymbol{t}^{(n)} - \hat{\boldsymbol{\mu}}). \tag{3.6}$$

Modelling a dataset with a Gaussian density is of course restrictive as only unimodal densities can be adequately modelled. This issue can be addressed by employing a mixture of Gaussians.

3.1.2 Multimodal Densities - Mixture of Gaussians

A mixture of C Gaussian densities is a composite model that comprises of Gaussian components $p(\boldsymbol{t}|\boldsymbol{\theta}_c)$ indexed by $c = 1, \ldots, C$:

$$p(\boldsymbol{t}|\boldsymbol{\theta}_c) = \mathcal{N}(\boldsymbol{t}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c). \tag{3.7}$$

We simplify the notation of model components $p(\boldsymbol{t}|\boldsymbol{\theta}_c)$ to $p(\boldsymbol{t}|c)$. The mixture is expressed as a linear combination of such components:

$$p(\boldsymbol{t}|\boldsymbol{\Theta}) = \sum_{c=1}^{C} P(c)p(\boldsymbol{t}|c). \tag{3.8}$$

Here $\boldsymbol{\Theta}$ is a vector that summarises the set of all parameters of the model, $\boldsymbol{\Theta} = \{P(1), \ldots, P(c), \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_c\}$. Quantities $P(c)$ are mixing coefficients subject to constraints $\sum_{c=1}^{C} P(c) = 1$ and $0 \leq P(c) \leq 1$. The log-likelihood reads:

$$\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) = \sum_{n=1}^{N} \log \sum_{c=1}^{C} P(c)p(\boldsymbol{t}^{(n)}|c). \tag{3.9}$$

As in the case of a single Gaussian, we could take derivatives of the log-likelihood with respect to parameters $\boldsymbol{\Theta}$ of the model to optimise it. However, a more elegant approach exists. Suppose we knew which component generated each data item. Let us express this knowledge by postulating a set of variables z that act as indicators of the origin of data items [Bishop, 1996]:

$$z_c^{(n)} = \begin{cases} 1, & \text{if } \boldsymbol{t}^{(n)} \text{ was generated by component } c; \\ 0, & \text{otherwise.} \end{cases} \tag{3.10}$$

We refer to the set of all such variables z by $\mathcal{Z}$. In light of such information, the log-likelihood in (3.9) is rewritten as:

$$\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z}) = \sum_{n=1}^{N} \sum_{c=1}^{C} z_c^{(n)} \log P(c)p(\boldsymbol{t}^{(n)}|c). \tag{3.11}$$

The parameters of each component c could then be independently estimated by taking into consideration only the data items that component c is responsible for generating:

$$P(c) = \frac{\sum_{i=1}^{N} z_c^{(i)}}{N}, \tag{3.12}$$

$$\boldsymbol{\mu}_c = \frac{1}{(\sum_{i=1}^{N} z_c^{(i)})} \sum_{n=1}^{N} z_c^{(n)} \boldsymbol{t}^{(n)}, \tag{3.13}$$

$$\boldsymbol{\Sigma}_C = \frac{1}{(\sum_{i=1}^{N} z_c^{(i)})} \sum_{n=1}^{N} z_c^{(n)} (\boldsymbol{t}^{(n)} - \boldsymbol{\mu}_c)^T (\boldsymbol{t}^{(n)} - \boldsymbol{\mu}_c). \tag{3.14}$$

Unfortunately in practice, information on variables z is not available. Nevertheless, it is possible to take advantage of such hidden information of the problem via the EM algorithm that treats variables z as random variables.

3.2 Overview of the Expectation-Maximisation Algorithm

The EM algorithm is an iterative optimisation technique for obtaining maximum-likelihood and maximum a-posteriori estimates in problems where certain information is unobserved. The information may be unobserved either because certain data values are missing in the dataset or because it is hidden from us, that is it cannot be directly measured. In the previous section we encountered the case of hidden information in the form of the $\mathcal{Z}$ variables. Here we shall consider the EM for training a mixture model, so again we postulate variables z to express hidden information on the component origin of each data item. Hence, we repeat:

$$z_c^{(n)} = \begin{cases} 1, & \text{component } c \text{ generated the } n\text{-th data item;} \\ 0, & \text{otherwise.} \end{cases} \tag{3.15}$$

We consider a dataset $\mathcal{D} = \{\boldsymbol{t}^{(1)}, ..., \boldsymbol{t}^{(N)}\}$, where $\boldsymbol{t}^{(n)}$ are vectors in $\mathbb{R}^d$, independently and identically distributed. Information of just $\mathcal{D}$ is termed as incomplete since certain knowledge is stored in the hidden variables $\mathcal{Z}$. The complete information is $(\mathcal{D}, \mathcal{Z})$. In the previous section, we saw that in the presence of complete information, the parameters of the mixture model could be straightforwardly calculated.

Let us consider a general mixture model with C components and parameters $\boldsymbol{\Theta}$ (we do not specify the form of the components). The log-likelihood is written as:

$$\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) = \sum_{n=1}^{N} \log \sum_{c=1}^{C} P(c) p(\boldsymbol{t}^{(n)}|c). \tag{3.16}$$

This expression is termed as the *incomplete-data log-likelihood*. It expresses the occurrence of the dataset $\mathcal{D}$ in the absence of the hidden information $\mathcal{Z}$. If we incorporate the hidden information

in $\mathcal{Z}$, we obtain a new expression for the log-likelihood:

$$\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z}) = \log p(\mathcal{D}, \mathcal{Z}|\boldsymbol{\Theta}) = \sum_{n=1}^{N} \sum_{c=1}^{C} z_c^{(n)} \log P(c) p(\boldsymbol{t}^{(n)}|c). \tag{3.17}$$

This expression is called the *complete-data log-likelihood* [Bilmes, 1997]. Since the hidden variables are random, $\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z})$ is also a random quantity. Although variables z are hidden from us, their underlying distribution can be estimated indirectly. This estimation relies on dataset $\mathcal{D}$ and the current model parameters, let us say $\boldsymbol{\Theta}^{(i)}$, i.e. model parameters at the i-th iteration of estimation process. An initial guess of the model parameters $\boldsymbol{\Theta}^{(1)}$ at iteration 1, can be either obtained by picking $\boldsymbol{\Theta}^{(1)}$ randomly from the corresponding domain of parameters or by some problem-specific initialisation procedure. What we are capable of calculating directly, are the conditional densities $p(\boldsymbol{t}|c, \boldsymbol{\Theta}^{(i)})$ for each component c, determined by model parameters $\boldsymbol{\Theta}^{(i)}$ at the i-th iteration. Using Bayes' theorem it is possible to obtain posterior probabilities:

$$p(c|\boldsymbol{t}, \boldsymbol{\Theta}^{(i)}) = \frac{P(c) p(\boldsymbol{t}|c, \boldsymbol{\Theta}^{(i)})}{p(\boldsymbol{t})} = \frac{P(c) p(\boldsymbol{t}|c, \boldsymbol{\Theta}^{(i)})}{\sum_{c'}^{C} P(c') p(\boldsymbol{t}|c', \boldsymbol{\Theta}^{(i)})}. \tag{3.18}$$

These posterior probabilities are necessary in calculating the conditional expectation of hidden variable $z_c^{(n)}$:

$$\begin{aligned}
E[z_c^{(n)}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] &= (z_c^{(n)} = 0) \times p(\boldsymbol{t}^{(n)} \text{ not generated by component } c) \\
&+ (z_c^{(n)} = 1) \times p(\boldsymbol{t}^{(n)} \text{ generated by component } c) \\
&= (z_c^{(n)} = 0) \times p(z_c^{(n)} = 0|\boldsymbol{\Theta}^{(i)}) + (z_c^{(n)} = 1) \times p(z_c^{(n)} = 1|\boldsymbol{\Theta}^{(i)}) \\
&= 0 \times p(\neg c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) + 1 \times p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \\
&= p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}).
\end{aligned} \tag{3.19}$$

The conditional expectation q of variables z given the dataset and the current model parameters $\boldsymbol{\Theta}^{(i)}$ can be written as:

$$q(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}) = q(\underbrace{z_1^{(1)}, z_2^{(1)}, ..., z_C^{(1)}}_{1}, \underbrace{z_1^{(2)}, z_2^{(2)}, ..., z_C^{(2)}}_{2}, ..., \underbrace{z_1^{(N)}, z_2^{(N)}, ..., z_C^{(N)}}_{N} |\mathcal{D}, \boldsymbol{\Theta}^{(i)}), \tag{3.20}$$

where the braces identify N subsets of z variables that concern the same data item. We assume that these N subsets of variables are independent from each other, given the dataset and the current parameters; knowing the values of any of the N subsets, does not convey any information about the values of any other subset. To rephrase: knowing that a data item originated from a

particular component, does not shed any light on the origin of some other data item. Hence:

$$\begin{aligned} q(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}) &= q_1(z_1^{(1)}, z_2^{(1)}, \ldots, z_C^{(1)}|\boldsymbol{t}^{(1)}, \boldsymbol{\Theta}^{(i)}) q_2(z_1^{(2)}, z_2^{(2)}, \ldots, z_C^{(2)}|\boldsymbol{t}^{(2)}, \boldsymbol{\Theta}^{(i)}) \times \ldots \\ &\times \quad q_N(z_1^{(N)}, z_2^{(N)}, \ldots, z_C^{(N)}|\boldsymbol{t}^{(N)}, \boldsymbol{\Theta}^{(i)}). \end{aligned} \tag{3.21}$$

Taking into consideration that within each of the N subsets of z variables, only one variable is equal to 1 (and the rest are equal to 0), the j-th factor of the conditional expectation in (3.21) can be rewritten using (3.19):

$$q_j(z_1^{(j)}, z_2^{(j)}, \ldots, z_C^{(j)}|\boldsymbol{t}^{(j)}, \boldsymbol{\Theta}^{(i)}) = \sum_{c=1}^{C} z_c^{(j)} p(c|\boldsymbol{t}^{(j)}, \boldsymbol{\Theta}^{(i)}). \tag{3.22}$$

This helps rewrite the condition expectation q as:

$$q(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}) = \prod_{n=1}^{N} \sum_{c=1}^{C} z_c^{(n)} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}). \tag{3.23}$$

Since expression (3.17) is random and cannot be maximised as it is, we could instead attempt to maximise its expectation with respect to the q distribution. Intuitively, this provides an estimate for the model parameters weighted by how likely each instance

$$\boldsymbol{z} = (z_1^{(1)}, \ldots, z_C^{(1)}, z_1^{(2)}, \ldots, z_C^{(2)}, \ldots, z_1^{(N)}, \ldots, z_C^{(N)})$$

of the hidden variables is. This leads us to the following expression:

$$\begin{aligned} E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] &= \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \sum_{n=1}^{N} \sum_{c=1}^{C} z_c^{(n)} \log[p(c) P(\boldsymbol{t}^{(n)}|c)] \\ &= \sum_{n=1}^{N} \sum_{c=1}^{C} \sum_{\boldsymbol{z}} q(\boldsymbol{z}) z_c^{(n)} \log[p(c) P(\boldsymbol{t}^{(n)}|c)]. \end{aligned} \tag{3.24}$$

Intuitively, the term $\sum_{\boldsymbol{z}} q(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}) z_c^{(n)}$ expresses the expectation of variable $z_c^{(n)}$ and must be

equal to $p(z_c^{(n)})$. We can see this by the following consideration using (3.23):

$$\begin{aligned}
\sum_{\boldsymbol{z}} q(\boldsymbol{z}|\mathcal{D},\boldsymbol{\Theta}^{(i)}) z_c^{(n)} &= \sum_{\boldsymbol{z}} \prod_{r=1}^{N}\sum_{s=1}^{C} z_s^{(r)} p(s|\boldsymbol{t}^{(r)},\boldsymbol{\Theta}^{(i)}) z_c^{(n)} \\
&= \sum_{\boldsymbol{z}} z_c^{(n)} \Big(\sum_{s=1}^{C} z_s^{(n)} p(s|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})\Big)\Big(\prod_{r=1,r\neq n}^{N}\sum_{s=1}^{C} z_s^{(r)} p(s|\boldsymbol{t}^{(r)},\boldsymbol{\Theta}^{(i)})\Big) \\
&= \sum_{\boldsymbol{z}} p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})\Big(\prod_{r=1,r\neq n}^{N}\sum_{s=1}^{C} z_s^{(r)} p(s|\boldsymbol{t}^{(r)},\boldsymbol{\Theta}^{(i)})\Big) \\
&= p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)}) \sum_{\boldsymbol{z}} \prod_{r=1,r\neq n}^{N}\sum_{s=1}^{C} z_s^{(r)} p(s|\boldsymbol{t}^{(r)},\boldsymbol{\Theta}^{(i)}) \\
&= p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)}) \frac{\sum_{\boldsymbol{z}} q(\boldsymbol{z}|\mathcal{D},\boldsymbol{\Theta}^{(i)})}{\sum_{\boldsymbol{z}} q_n(\boldsymbol{z}|\mathcal{D},\boldsymbol{\Theta}^{(i)})} \\
&= p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)}) = p(z_c^{(n)}). \qquad (3.25)
\end{aligned}$$

The result obtained by taking into consideration that $\sum_{\boldsymbol{z}} q(\boldsymbol{z}|\mathcal{D},\boldsymbol{\Theta}^{(i)}) = 1$ and that:

$$\begin{aligned}
\sum_{\boldsymbol{z}} q_n(\boldsymbol{z}|\mathcal{D},\boldsymbol{\Theta}^{(i)}) &= \sum_{\boldsymbol{z}}\sum_{c=1}^{C} z_c^{(n)} p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)}) = \sum_{c=1}^{C}\sum_{\boldsymbol{z}} z_c^{(n)} p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)}) \\
&= \sum_{c=1}^{C}\Big(p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})\sum_{\boldsymbol{z}} z_c^{(n)}\Big) = \sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)}) = 1. \qquad (3.26)
\end{aligned}$$

Substituting this in (3.24) yields:

$$E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\Theta}^{(i)}] = \sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)}) \log[P(c)p(\boldsymbol{t}^{(n)}|c)]. \qquad (3.27)$$

This expression is called the *expected complete-data log-likelihood*. It is a deterministic function of $\boldsymbol{\Theta}$. Note that $\boldsymbol{\Theta}^{(i)}$ refers to the parameters used to estimate distribution $p(\boldsymbol{z}|\mathcal{D},\boldsymbol{\Theta}^{(i)})$ and evaluate the expectation, while $\boldsymbol{\Theta}$ are the free variables. Why is this expression useful? Our original goal was after all to maximise the model log-likelihood $\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D})$. One may view EM as a lower bound maximisation algorithm. That is, instead of optimising $\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D})$, it optimises an auxiliary function. This auxiliary function is a lower bound of $\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D})$ that can be made "tight". Maximising the lower bound, effectively "pushes" $\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D})$ "upwards". Such a lower bound can be derived by starting from the model log-likelihood and introducing a

distribution $q(\boldsymbol{z})$ for the variables in $\mathcal{Z}$ as follows ([Salakhutdinov et al., 2003, Minka, 1998]):

$$\begin{aligned}\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) &= \log p(\mathcal{D}|\boldsymbol{\Theta}) = \log \sum_{\boldsymbol{z}} p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\Theta}) \\ &= \log \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\Theta})}{q(\boldsymbol{z})} \\ &\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\Theta})}{q(\boldsymbol{z})}. \end{aligned} \tag{3.28}$$

The inequality in the last line is obtained from *Jensen's inequality* [Cover and Thomas, 1991] for convex functions. It provides a lower bound on $\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D})$ for any arbitrary, positive distribution q. This lower bound is the auxiliary function that we optimise instead. We define the auxiliary function as a function F:

$$F(q, \boldsymbol{\Theta}) = \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\Theta})}{q(\boldsymbol{z})}. \tag{3.29}$$

Auxiliary function $F(q, \boldsymbol{\Theta})$ is a function of two parameters, the distribution q of hidden variables z and model parameters $\boldsymbol{\Theta}$. The EM algorithm optimises function F in a coordinate-wise fashion; it alternates between optimising q while keeping $\boldsymbol{\Theta}$ fixed to the current parameters $\boldsymbol{\Theta}^{(i)}$ and then optimising $\boldsymbol{\Theta}$ while keeping q fixed to the values attained previously. In order to optimise for q, we rewrite F as:

$$\begin{aligned} F(q, \boldsymbol{\Theta}^{(i)}) &= \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}) p(\mathcal{D}, \boldsymbol{\Theta}^{(i)})}{q(\boldsymbol{z})} \\ &= -\sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)})} + \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log p(\mathcal{D}|\boldsymbol{\Theta}^{(i)}) \\ &= -D_{KL}[q(\boldsymbol{z})||p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)})] + const, \end{aligned} \tag{3.30}$$

where in the second line we discard the second summand being a constant, and $D_{KL}[q(\boldsymbol{z})||p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)})]$ is the *Kullback-Leibler* divergence (KLD) [Cover and Thomas, 1991] between the two distributions $q(\boldsymbol{z})$ and $p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)})$. KLD is a non-negative scalar quantity that informs us of the "distance" between two distributions. In order to maximise the non-negative KLD quantity here, we simply set $q(\boldsymbol{z}) = p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)})$. Thus, at each iteration, setting q involves the estimation of the posterior distribution of $\boldsymbol{z}$ conditioned on the dataset $\mathcal{D}$ and the current parameters $\boldsymbol{\Theta}^{(i)}$. This step is called the *Estimation step* or *E-step*. Having determined $q(\boldsymbol{z}) = p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)})$,

we proceed to the optimisation of $\boldsymbol{\Theta}$. To this purpose, we rewrite F as follows:

$$\begin{aligned} F(p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}), \boldsymbol{\Theta}) &= \sum_{\boldsymbol{z}} p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}) \log \frac{p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\Theta})}{p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)})} \\ &= \sum_{\boldsymbol{z}} p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}) \log p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\Theta}) - \sum_{\boldsymbol{z}} p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}) \log p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}) \\ &= E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] - const. \end{aligned} \tag{3.31}$$

We see that F is equal to the expected complete-data log-likelihood in (3.27) plus a constant entropy term that can be discarded in the optimisation of $\boldsymbol{\Theta}$ since it is independent from it. The step of optimising $\boldsymbol{\Theta}$ is called the *Maximisation step* or *M-step*. As aforementioned, the expected complete-data log-likelihood is a deterministic function of $\boldsymbol{\Theta}$ and can be optimised by a variety of optimisation techniques.

To summarise, the EM algorithm maximises indirectly the model log-likelihood via an auxiliary function that acts as a lower bound:

$$\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) \geq F(q, \boldsymbol{\Theta}). \tag{3.32}$$

The auxiliary function has two arguments, q the conditional expectation of the hidden variables, and $\boldsymbol{\Theta}$ the model parameters, that are optimised in a coordinate-wise fashion. This gives forth to the following two steps:

- E-step: estimate the distribution of variables z given the dataset $\mathcal{D}$ and current model parameters $\boldsymbol{\Theta}^{(i)}$:
$$p(z_c^{(n)}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}) = \frac{P(c)p(\boldsymbol{t}^{(n)}|c, \boldsymbol{\Theta}^{(i)})}{\sum_{c'}^{C} P(c')p(\boldsymbol{t}^{(n)}|c', \boldsymbol{\Theta}^{(i)})}.$$
- M-step: optimise the expected complete-data log-likelihood as a function of $\boldsymbol{\Theta}$:
$$\boldsymbol{\Theta}^{(i+1)} = \underset{\boldsymbol{\Theta}}{argmax} \left\{ E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] \right\}.$$

In the case of MAP estimation, where a prior $p(\boldsymbol{\Theta})$ is imposed on the model parameters $\boldsymbol{\Theta}$, the model log-likelihood must account for an additional term:

$$\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) = \log p(\boldsymbol{\Theta}) + \sum_{n=1}^{N} \log \sum_{c=1}^{C} P(c)p(\boldsymbol{t}^{(n)}|c). \tag{3.33}$$

The EM methodology changes only slightly ([Ng et al., 2004]); the E-step remains as is, while the M-step now must take into consideration the simultaneous optimisation of the log-prior

term:

$$\boldsymbol{\Theta}^{(i+1)} = \underset{\boldsymbol{\Theta}}{argmax} \left\{ \log p(\boldsymbol{\Theta}) + E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] \right\}. \tag{3.34}$$

The next section presents the E- and M-step for training a mixture of Gaussians.

3.2.1 Training of Mixture of Gaussians

Let us return to the mixture of Gaussians. We state again the model log-likelihood:

$$\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) = \sum_{n=1}^{N} \log \sum_{c=1}^{C} P(c) p(\boldsymbol{t}^{(n)}|c). \tag{3.35}$$

Instead of directly maximising the model log-likelihood, we follow the EM methodology and maximise the expected complete-data log-likelihood $E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\Theta}^{(i)}]$ that acts as a lower bound. To that purpose, we must estimate the conditional expectation q of the hidden variables z given dataset $\mathcal{D}$ and model parameters $\boldsymbol{\Theta}^{(i)}$ at the E-step of the i-th iteration:

$$E[z_c^{(n)}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] = p(z_c^{(n)} = 1|\boldsymbol{\Theta}^{(i)}) = p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) = \frac{P(c) p(\boldsymbol{t}^{(n)}|c, \boldsymbol{\Theta}^{(i)})}{\sum_{c'=1}^{C} P(c') p(\boldsymbol{t}^{(n)}|c', \boldsymbol{\Theta}^{(i)})}. \tag{3.36}$$

Once q is established, we can calculate the expected complete-data log-likelihood in (3.27). We now improve the bound by maximising $\boldsymbol{\Theta}$. This is done by taking derivatives with respect to the parameters of the model and setting them to zero. Following the derivation in [Bishop, 1996, Bilmes, 1997], we obtain the following update equations for the M-step of i-th iteration:

$$P(c) = \frac{1}{N} \sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}), \tag{3.37}$$

$$\boldsymbol{\mu}_c = \frac{\sum_{n=1}^{N} \boldsymbol{t}^{(n)} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})}{\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})}, \tag{3.38}$$

$$\boldsymbol{\Sigma}_c = \frac{\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})(\boldsymbol{t}^{(n)} - \mu_c)^T (\boldsymbol{t}^{(n)} - \mu_c)}{\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})}. \tag{3.39}$$

These equations make intuitive sense [Bishop, 1999] and should be compared to (3.5) and (3.6). The posterior probability $p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})$ can be interpreted as the *responsibility* of each component c giving rise to data item $\boldsymbol{t}^{(n)}$. Thus, if component c is highly responsible, that is the posterior probability $p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})$ is high, the contribution of data item $\boldsymbol{t}^{(n)}$ in updating the parameters of component c will be high. Conversely, low responsibilities weigh the contribution

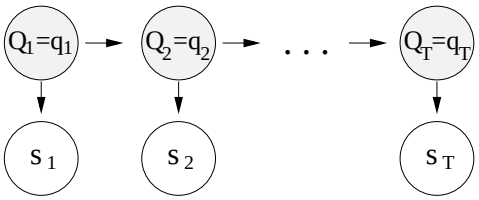


Fig. 3.1: Example of an underlying hidden state (states in gray) process emitting labels.

of the corresponding data items lowly.

Even though, we introduced the EM algorithm in the specific setting of finding parameters of a mixture of Gaussian densities, the EM has a wider range of applications. In the following sections the EM is employed in training hidden Markov models and hidden Markov tree models.

3.3 Modelling Sequences

3.3.1 Overview of Hidden Markov Models

We denote the domain of sequences by $\mathcal{S}$. A variable over $\mathcal{S}$ is denoted by $\boldsymbol{S}$. We assume that all sequences are of equal length T and we index symbols of a sequence by $t = 1, ..., T$. Thus, a sequence $\boldsymbol{S}$ is expressed as $\boldsymbol{S} = [\boldsymbol{S}_1 \boldsymbol{S}_2 \dots \boldsymbol{S}_T]$. Sequences may be composed of symbols that belong either to a discrete or continuous domain. In the first case symbols belong to a fixed alphabet $\mathcal{A}$ with A number of elements, while in the second case to a domain $\mathbb{R}^d$. An instantiation of a sequence is expressed as $\boldsymbol{S} = \boldsymbol{s} = [(\boldsymbol{S}_1 = \boldsymbol{s}_1)(\boldsymbol{S}_2 = \boldsymbol{s}_2) \dots (\boldsymbol{S}_1 = \boldsymbol{s}_1)]$.

Sequences can be modelled by a *hidden Markov model* (HMM) [Rabiner, 1989]. A HMM defines a discrete random variable Q_t for each symbol $\boldsymbol{S}_t, t = 1, \dots, T$ of a sequence $\boldsymbol{S}$. The variables can be in one of K discrete unobservable states, $Q_t = \{1, 2, \dots, K\}$. Each state variable Q_t emits a symbol $\boldsymbol{S}_t$. This is illustrated in Fig. 3.1. What symbol will be emitted depends on the particular state $Q_t = q_t$ that is entered at step t. In particular, for discrete symbols the emission is governed by a $K \times A$ matrix $\boldsymbol{\Psi}$, where the k, j entry stands for the probability of emitting symbol j at state k, that is $\psi_{kj} = p(\boldsymbol{s}_t = j | Q_t = k)$. For the continuous domain each state k is associated with a probability density $f(\cdot; \boldsymbol{\psi}_k)$, where $\boldsymbol{\psi}_k$ is a vector of parameters for state k. As one expects in a sequence, symbols $\boldsymbol{S}_t$ are not independent of each other. A HMM captures this property of sequences by making a state Q_t dependent on its preceding state Q_{t-1}. Transitions between states are governed by a $K \times K$ transition matrix $\boldsymbol{B}$ with elements $b_{kl} = p(Q_t = l | Q_{t-1} = k)$ with $k, l = 1, \dots, K$. Note that the sum of elements b_{kl} over $l = 1, \dots, K$, i.e. the sum of transitions from a state k to all other K states, must equal 1.

For the very first state Q_1 we define a vector $\boldsymbol{\pi}$ of initial probabilities that governs the first

state entered, with elements $\pi_k = p(Q_1 = k)$ with $k = 1, \ldots, K$. We group the set of the preceding HMM parameters in parameter vector $\boldsymbol{\theta}$.

A HMM implements a *first-order Markov* property by taking into account the following conditional independencies [Bilmes, 1997]:

- A state Q_t is independent from all other previous variables given its direct state predecessor Q_{t-1}:

$$p(Q_t = q_t | \{Q_r = q_r\}_{r=1,\ldots,t-1}, \{\boldsymbol{S}_r = \boldsymbol{s}_r\}_{r=1,\ldots,t-1}) = p(Q_t = q_t | Q_t = q_{t-1}). \tag{3.40}$$

- A symbol $\boldsymbol{S}_t$ is independent of all other variables given its state Q_t:

$$p(\boldsymbol{S}_t = \boldsymbol{s}_t | \{\boldsymbol{S}_r = \boldsymbol{s}_r\}_{r=1,\ldots,T, r \neq t}, \{Q_r = q_r\}_{r=1,\ldots,T}) = p(\boldsymbol{S}_t = \boldsymbol{s}_t | Q_{t-1} = q_{t-1}). \tag{3.41}$$

Thus, a HMM can be factorised as follows:

$$\begin{aligned} p(\boldsymbol{S} = \boldsymbol{s}, Q_1 = q_1, \ldots, Q_T = q_T | \boldsymbol{\theta}) &= p(Q_1 = q_1 | \boldsymbol{\theta}) \prod_{t=2}^{T} p(Q_t = q_t | Q_{t-1} = q_{t-1}, \boldsymbol{\theta}) \\ &\times \prod_{t=1}^{T} p(\boldsymbol{S}_t = \boldsymbol{s}_t | Q_t = q_t, \boldsymbol{\theta}). \end{aligned} \tag{3.42}$$

Henceforth, for brevity we shall drop stating both random variables and their instantiations, keeping only the latter, i.e. $p(\boldsymbol{S}_t = \boldsymbol{s}_t | Q_t = q_t, \boldsymbol{\theta}) = p(\boldsymbol{s}_t | q_t, \boldsymbol{\theta})$. To compute the likelihood of a HMM given a sequence, the forward-backward algorithm is employed. The motivation for using this algorithm, stems from the observation that a direct calculation of the likelihood requires an exponential number of steps:

$$\begin{aligned} p(\boldsymbol{s} | \boldsymbol{\theta}) &= \sum_{\boldsymbol{q} \in \{1,2,\ldots,K\}^T} p(\boldsymbol{s}, \boldsymbol{q} | \boldsymbol{\theta}) = \sum_{q_1=1}^{K} \sum_{q_2=1}^{K} \cdots \sum_{q_T=1}^{K} p(\boldsymbol{s}, q_1, \ldots, q_T | \boldsymbol{\theta}) \\ &= \sum_{q_1=1}^{K} \sum_{q_2=1}^{K} \cdots \sum_{q_T=1}^{K} p(q_1 | \boldsymbol{\theta}) p(\boldsymbol{s}_1 | q_1, \boldsymbol{\theta}) p(q_2 | q_1, \boldsymbol{\theta}) \ldots p(\boldsymbol{s}_T | q_T, \boldsymbol{\theta}) p(q_T | q_{T-1}, \boldsymbol{\theta}), \end{aligned} \tag{3.43}$$

where vector $\boldsymbol{q}$ refers to a configuration of states $\boldsymbol{q} = [q_1 q_2 ... q_T]$. In [Bilmes, 1997, Rabiner, 1989] the forward and backward algorithms are presented as efficient recursions for calculating the likelihood. First we introduce the following quantity, called the forward probability:

$$\alpha_k(t; \boldsymbol{\theta}) = p(\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_t, q_t = k | \boldsymbol{\theta}), \tag{3.44}$$

which is the probability of observing sequence $\boldsymbol{s}$ up to step t where the state is k. Based on this, we calculate the likelihood in the following recursive way:

- Initial step, starts from the beginning of the sequence:

$$\alpha_k(1;\boldsymbol{\theta}) = p(\boldsymbol{s}_1, q_1 = k|\boldsymbol{\theta}) = p(\boldsymbol{s}_1|q_1 = k, \boldsymbol{\theta})p(q_1 = k|\boldsymbol{\theta}). \tag{3.45}$$

- Recursive step:

$$\alpha_k(t;\boldsymbol{\theta}) = \left[\sum_{l=1}^{K} \alpha_l(t-1;\boldsymbol{\theta})p(q_t = k|q_{t-1} = l, \boldsymbol{\theta})\right] p(\boldsymbol{s}_t|q_t = k, \boldsymbol{\theta}). \tag{3.46}$$

- Final step, the likelihood is calculated as:

$$p(\boldsymbol{s}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \alpha_k(T;\boldsymbol{\theta}). \tag{3.47}$$

In the same fashion we derive the backward algorithm, here we introduce the backward probability:

$$\beta_k(t;\boldsymbol{\theta}) = p(\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+2}, \ldots, \boldsymbol{s}_T|q_t = k, \boldsymbol{\theta}), \tag{3.48}$$

which is the probability of observing sequence $\boldsymbol{s}$ from time step $t+1$ onwards given that the state at step t is k. Based on this we can calculate the likelihood in the following recursive way:

- Initial step, starts from the end of the sequence:

$$\beta_k(T;\boldsymbol{\theta}) = p(\boldsymbol{s}_T|q_T = k, \boldsymbol{\theta}). \tag{3.49}$$

- Recursive step:

$$\beta_k(t;\boldsymbol{\theta}) = p(\boldsymbol{s}_t|q_t = k, \boldsymbol{\theta}) \sum_{l=1}^{K} p(q_{t+1} = l|q_t = k, \boldsymbol{\theta})\beta_l(t+1;\boldsymbol{\theta}). \tag{3.50}$$

- Final step, the likelihood is calculated as:

$$p(\boldsymbol{s}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \alpha_k(u;\boldsymbol{\theta})\beta_k(u;\boldsymbol{\theta}). \tag{3.51}$$

3.3.2 Training of Hidden Markov Models

Here we consider a dataset $\mathcal{D} = \left\{\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)}, \ldots, \boldsymbol{s}^{(N)}\right\}$ of independently generated sequences. The model likelihood given dataset $\mathcal{D}$ is expressed as:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) &= \prod_{n=1}^{N} p(\boldsymbol{s}^{(n)}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \sum_{\boldsymbol{q}\in\{1,2,\ldots,K\}^T} p(q_1|\boldsymbol{\theta}) \prod_{t=2}^{T} p(q_t|q_{t-1},\boldsymbol{\theta}) \\ &\times \prod_{t=1}^{T} p(\boldsymbol{s}_t^{(n)}|q_t,\boldsymbol{\theta}). \end{aligned} \tag{3.52}$$

We postulate the following hidden indicator variables:

$$z_k^{(n,t)} = \begin{cases} 1, & \text{if for } \boldsymbol{s}^{(n)} \text{ at step } t, \text{ the state was } q_t = k; \\ 0, & \text{otherwise.} \end{cases}$$

$$z_{k\to l}^{(n,t)} = \begin{cases} 1, & \text{if for } \boldsymbol{s}^{(n)} \text{ at step } t-1, \text{ the state was } q_{t-1} = k \\ & \text{and at step } t \text{ the state was } q_t = l; \\ 0, & \text{otherwise.} \end{cases}$$

Again, we refer to all variables z by $\mathcal{Z}$. Training of HMMs proceeds via the EM algorithm. The EM algorithm alternates between the E-step, where the expectation of the hidden variables at the i-th iteration is estimated based on the dataset $\mathcal{D}$ and the current parameters $\boldsymbol{\theta}^{(i)}$, and the M-step where the model parameters $\boldsymbol{\theta}$ are updated by optimising the expected complete-data log-likelihood. Using the above indicator variables and based on (3.42), we rewrite the model likelihood and take the logarithm, which leads us to the complete-data log-likelihood function as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\mathcal{D},\mathcal{Z}) &= \prod_{n=1}^{N} p(\boldsymbol{s}^{(n)}|\boldsymbol{\theta}) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(q_1=k|\boldsymbol{\theta})^{z_k^{(n,1)}} \prod_{t=2}^{T}\prod_{k=1}^{K}\prod_{l=1}^{K} p(q_t=k|q_{t-1}=l,\boldsymbol{\theta})^{z_{k\to l}^{(n,t)}} \\ &\times \prod_{t=1}^{T}\prod_{k=1}^{K} p(\boldsymbol{s}_t^{(n)}|q_t=k,\boldsymbol{\theta})^{z_k^{(n,t)}}, \end{aligned} \tag{3.53}$$

$$\begin{aligned}\log\mathcal{L}(\boldsymbol{\theta}|\mathcal{D},\mathcal{Z}) &= \sum_{n=1}^{N}\Bigg(\sum_{k=1}^{K} z_k^{(n,1)} \log p(q_1=k|\boldsymbol{\theta}) + \sum_{t=2}^{T}\sum_{k=1}^{K}\sum_{l=1}^{K} z_{k\to l}^{(n,t)} \log p(q_t=k|q_{t-1}=l,\boldsymbol{\theta}) \\ &+ \sum_{t=1}^{T}\sum_{k=1}^{K} z_k^{(n,t)} \log p(\boldsymbol{s}_t^{(n)}|q_t=k,\boldsymbol{\theta})\Bigg). \end{aligned} \tag{3.54}$$

We estimate the expectation of hidden variables z in the E-step, based on the current model parameters $\boldsymbol{\theta}^{(i)}$. Estimation proceeds with the aid of the forward-backward probabil-

ities [Rabiner, 1989]:

$$\begin{aligned} E[z_k^{(n,t)}|\mathcal{D},\boldsymbol{\theta}^{(i)}] &= p(q_t=k|\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)}) = \frac{p(q_t=k,\boldsymbol{s}^{(n)}|\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{s}^{(n)}|\boldsymbol{\theta}^{(i)})} = \frac{p(q_t=k,\boldsymbol{s}^{(n)}|\boldsymbol{\theta}^{(i)})}{\sum_{j=1}^{K} p(q_t=l,\boldsymbol{s}^{(n)}|\boldsymbol{\theta}^{(i)})} \\ &= \frac{\alpha_k^{(n)}(t;\boldsymbol{\theta}^{(i)})\beta_k^{(n)}(t;\boldsymbol{\theta}^{(i)})}{\sum_{l=1}^{K}\alpha_l^{(n)}(t;\boldsymbol{\theta}^{(i)})\beta_l^{(n)}(t;\boldsymbol{\theta}^{(i)})}, \end{aligned} \tag{3.55}$$

$$\begin{aligned} E[z_{k\to l}^{(n,t)}|\mathcal{D},\boldsymbol{\theta}^{(i)}] &= p(q_t=l,q_{t-1}=k|\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)}) = \frac{p(q_t=l,q_{t-1}=k,\boldsymbol{s}^{(n)}|\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{s}^{(n)}|\boldsymbol{\theta}^{(i)})} \\ &= \frac{\alpha_k^{(n)}(t-1;\boldsymbol{\theta}^{(i)})p(q_t=l|q_{t-1}=k,\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)})p(\boldsymbol{s}_t^{(n)}|q_t=l,\boldsymbol{\theta}^{(i)})\beta_l^{(n)}(t;\boldsymbol{\theta}^{(i)})}{\sum_{k'=1}^{K}\sum_{l'=1}^{K}\alpha_{k'}^{(n)}(t-1;\boldsymbol{\theta}^{(i)})p(q_t=l'|q_{t-1}=k',\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)})p(\boldsymbol{s}_t^{(n)}|q_t=l',\boldsymbol{\theta}^{(i)})\beta_{l'}^{(n)}(t;\boldsymbol{\theta}^{(i)})}. \end{aligned} \tag{3.56}$$

where $\alpha_k^{(n)}(t;\boldsymbol{\theta}^{(i)}), \beta_k^{(n)}(t;\boldsymbol{\theta}^{(i)})$ are the forward and backward probabilities for the n-th sequence corresponding to state k respectively. Based on (3.54), the expected complete-data log-likelihood can now be written as:

$$\begin{aligned} E_{\mathcal{Z}}[\log\mathcal{L}(\boldsymbol{\theta}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\theta}^{(i)}] &= \sum_{n=1}^{N}\Bigg(\sum_{k=1}^{K} p(q_1=k|\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)})\log p(q_1=k|\boldsymbol{\theta}) \\ &+ \sum_{t=2}^{T}\sum_{k=1}^{K}\sum_{l=1}^{K} p(q_t=l,q_{t-1}=k|\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)})\log p(q_t=l|q_{t-1}=k,\boldsymbol{\theta}) \\ &+ \sum_{t=1}^{T}\sum_{k=1}^{K} p(q_t=k|\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)})\log p(\boldsymbol{s}_t^{(n)}|q_t=k,\boldsymbol{\theta})\Bigg). \end{aligned} \tag{3.57}$$

Calculating the derivatives of (3.57) with respect to the parameters of the model results in the following update equations [Bilmes, 1997] at the i-th iteration:

$$\pi_k = \frac{1}{N}\sum_{n=1}^{N} p(q_1=k|\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)}), \tag{3.58}$$

$$b_{kl} = \frac{\sum_{n=1}^{N}\sum_{t=2}^{T} p(q_t=l,q_{t-1}=k|\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)})}{\sum_{n=1}^{N}\sum_{t=2}^{T} p(q_{t-1}=k|\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)})}. \tag{3.59}$$

In the case of discrete symbols, the entries for matrix $\boldsymbol{\Psi}$ are updated as follows:

$$\psi_{kj} = \frac{\sum_{t=1}^{T}\delta_{j,\boldsymbol{s}_t} p(q_t=k|\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)})}{\sum_{t=1}^{T} p(q_t=k|\boldsymbol{s}^{(n)},\boldsymbol{\theta}^{(i)})}, \tag{3.60}$$

where δ is the Kronecker delta. If emission distribution $f(.;\boldsymbol{\psi}_k)$ assumes the form of a Gaussian,

then its parameters $\boldsymbol{\psi} = \{\boldsymbol{\psi}_k\}_{k=1,\ldots,K}$, with $\boldsymbol{\psi}_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ and $\boldsymbol{\mu}_k \in \mathbb{R}^d$, $\boldsymbol{\Sigma}_k \in \mathbb{R}^{d\times d}$ the mean and the covariance matrix for each state k respectively, are updated as follows:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T} \boldsymbol{s}_t^{(n)} p(q_t = k|\boldsymbol{s}^{(n)}, \boldsymbol{\theta}^{(i)})}{\sum_{n=1}^{N}\sum_{t=1}^{T} p(q_t = k|\boldsymbol{s}^{(n)}, \boldsymbol{\theta}^{(i)})}, \tag{3.61}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T} (\boldsymbol{s}_t^{(n)} - \boldsymbol{\mu}_k)^T(\boldsymbol{s}_t^{(n)} - \boldsymbol{\mu}_k) p(q_t = k|\boldsymbol{s}^{(n)}, \boldsymbol{\theta}^{(i)})}{\sum_{n=1}^{N}\sum_{t=1}^{T} p(q_t = k|\boldsymbol{s}^{(n)}, \boldsymbol{\theta}^{(i)})}. \tag{3.62}$$

The update equations have an intuitive interpretation similar to the update equations for the mixture of Gaussians in section 3.2.1. Quantity $p(q_t = k|\boldsymbol{s}^{(n)}, \boldsymbol{\theta}^{(i)})$ expresses the *responsibility* of state k giving rise to emission $\boldsymbol{s}_t^{(n)}$ at time t. When updating parameters that correspond to state k, the updates are weighted by the responsibilities so that: if state k is highly responsible, then the data item contributes highly; conversely, if state k is only weakly responsible, the data item contributes to a lesser degree.

3.3.3 Mixtures of Hidden Markov Models

Mixture of HMMs can be formulated in the same fashion as mixtures of Gaussian densities. Following the EM methodology in [Bilmes, 1997, Cadez et al., 2000], we heuristically demonstrate how such a mixture can be realised, based on the intuitive interpretation of the update equations that we saw in the training of mixture of Gaussians and HMMs.

Before proceeding further, however, we extend our notation with $c = 1, \ldots, K$ to index parameters that correspond to each of the C HMM components of the mixture model. Thus, each HMM component c has a non-negative mixing coefficient $P(c)$, with $\sum_{c=1}^{C} P(c) = 1$ and $0 \leq P(c) \leq 1$, an initial probability vector $\boldsymbol{\pi}^c = \{\pi_k^c\}_{k=1,\ldots,K}$, a transition probability matrix $\boldsymbol{B}^c$ with elements $b_{kl}^c = p(q_t = l|q_{t-1} = k, c)$, means for the emissions $\boldsymbol{\mu}_{c,k} \in \mathbb{R}^d$ with $k = 1, \ldots, K$ and covariances of the emissions $\boldsymbol{\Sigma}_{c,k} \in \mathbb{R}^{d\times d}$ with $k = 1, \ldots, K$.

We state the log-likelihood of the mixture model, for dataset $\mathcal{D}$:

$$\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) = \sum_{n=1}^{N} \log \sum_{c=1}^{C} P(c) p(\boldsymbol{s}^{(n)}|c). \tag{3.63}$$

We convert the problem into a hidden information problem by postulating anew the hidden variables with the addition of one extra set of variables that indicate the component-membership of each data item:

$$z_c^{(n)} = \begin{cases} 1, & \text{if } \boldsymbol{s}^{(n)} \text{ was generated by component } c; \\ 0, & \text{otherwise.} \end{cases}$$

$$z_{c,k}^{(n,t)} = \begin{cases} 1, & \text{given } \boldsymbol{s}^{(n)} \text{ was generated by component } c, \text{ if at step } t, \text{ the state was } q_t = k; \\ 0, & \text{otherwise.} \end{cases}$$

$$z_{c,k\to l}^{(n,t)} = \begin{cases} 1, & \text{given } \boldsymbol{s}^{(n)} \text{ was generated by component } c, \text{ if at step } t-1, \text{ the state was } q_{t-1} = k \\ & \text{and at step } t \text{ the state was } q_t = l; \\ 0, & \text{otherwise.} \end{cases}$$

The expectations of these hidden variables are respectively:

$$E[z_c^{(n)}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] = p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}), \tag{3.64}$$

$$E[z_{c,k}^{(n,t)}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] = p(q_t = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}), \tag{3.65}$$

$$E[z_{c,k\to l}^{(n,t)}|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] = p(q_t = l, q_{t-1} = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}). \tag{3.66}$$

We can now calculate the expected complete-data log-likelihood of the mixture model:

$$\begin{aligned} E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] &= \sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) \Bigg(\sum_{k=1}^{K} p(q_1 = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) \log p(q_1 = k|c, \boldsymbol{\Theta}) \\ &+ \sum_{t=2}^{T}\sum_{k=1}^{K}\sum_{l=1}^{K} p(q_t = l, q_{t-1} = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) \log p(q_t = l|q_{t-1} = k, c, \boldsymbol{\Theta}) \\ &+ \sum_{t=1}^{T}\sum_{k=1}^{K} p(q_t = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) \log p(\boldsymbol{s}_t^{(n)}|q_t = k, c, \boldsymbol{\Theta}) \Bigg). \end{aligned} \tag{3.67}$$

These expectations measure the contribution of each data item in updating the parameters of the mixture model. Thus, the mixing coefficients are updated in the same fashion as the mixing coefficients for the mixture of Gaussians:

$$P(c) = \frac{1}{N}\sum_{n=1}^{N} p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}). \tag{3.68}$$

The rest of the update equations are only slightly modified in order to calculate parameters specific for each component c:

$$\pi_{c,k} = \frac{1}{N}\sum_{n=1}^{N} p(q_1 = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}), \tag{3.69}$$

$$b_{c,kl} = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T} p(q_t = l, q_{t-1} = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)})}{\sum_{n=1}^{N}\sum_{t=1}^{T} p(q_{t-1} = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)})}, \tag{3.70}$$

$$\boldsymbol{\mu}_{c,k} = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T} \boldsymbol{s}_t^{(n)} p(q_t = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)})}{\sum_{n=1}^{N}\sum_{t=1}^{T} p(q_t = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)})}, \tag{3.71}$$

$$\boldsymbol{\Sigma}_{c,k} = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T} (\boldsymbol{s}_t^{(n)} - \boldsymbol{\mu}_k)^T (\boldsymbol{s}_t^{(n)} - \boldsymbol{\mu}_k) p(q_t = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)})}{\sum_{n=1}^{N}\sum_{t=1}^{T} p(q_t = k|c, \boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)}) p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)})}. \tag{3.72}$$

We observe that the update equations bear very close resemblance to the update equations when training a single HMM, the difference being that we now must also weigh the updates by the component contributions $p(c|\boldsymbol{s}^{(n)}, \boldsymbol{\Theta}^{(i)})$.

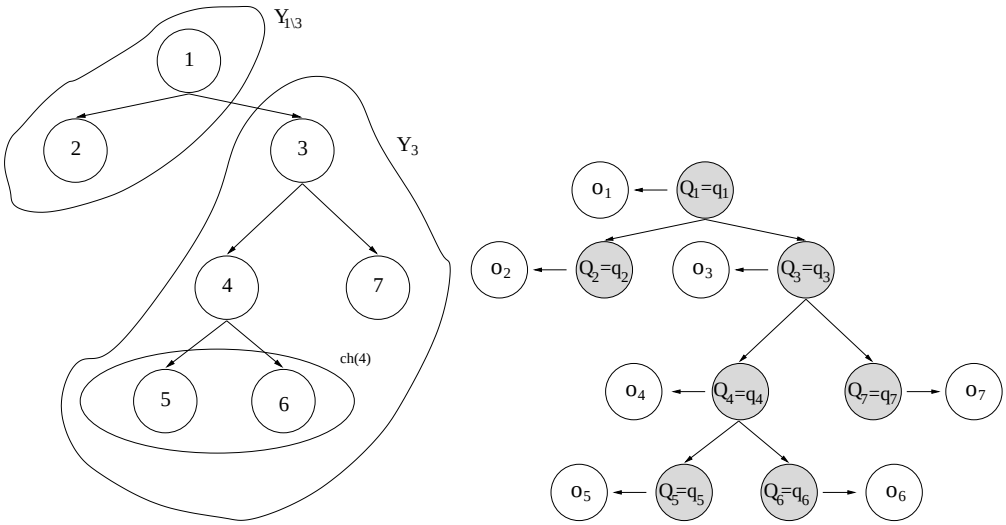
3.4 Modelling Tree Structures

3.4.1 Overview of Hidden Markov Tree Models

A tree $\boldsymbol{y}$ is an acyclic directed graph and as such it consists of a set $\mathcal{U}_{\boldsymbol{y}} = \{1, 2, ..., U_{\boldsymbol{y}}\}$ of nodes $u \in \mathcal{U}_{\boldsymbol{y}}$, a set of directed edges between the nodes (each edge goes from a parent node to a child node) and a set of labels [1] $\boldsymbol{o}_u \in \mathbb{R}^d$ on nodes $u \in \mathcal{U}_{\boldsymbol{y}}$. Each node u has a single parent $\rho(u)$ (apart from the node number one, the root node) and each node u has a set of children $ch(u)$ (apart from the leaf nodes). Furthermore we designate subtrees. A subtree rooted at node u of a tree $\boldsymbol{y}$ is referred to by $\boldsymbol{y}_u$. Hence the entire tree $\boldsymbol{y}$ is equivalent to the subtree $\boldsymbol{y}_1$ rooted at its root. Moreover, $\boldsymbol{y}_{u\backslash v}$ denotes the entire subtree $\boldsymbol{y}_u$ except for the subtree rooted at node v. This notation is illustrated in Fig. 3.2(a).

We also introduce a model for the labels of the trees that captures the structure of the trees. We associate with each node u a discrete random variable Q_u which can be in one of K unobservable states. The variable Q_u stochastically determines the label for node u. Each state $k = 1, 2, ..., K$ is associated with a parametrised emission distribution $f(.; \boldsymbol{\psi}_k)$ that produces a label. So given a tree structure $\boldsymbol{y}$, the model can label each of the nodes depending on what state $Q_u \in \{1, 2, ..., K\}$ each node u is in. What states will be entered and ultimately what labels will be produced depends on the structure of the model. By structure we mean a joint probability distribution over state variables that characterises the relationship between states Q_u of all nodes $u \in \mathcal{U}_{\boldsymbol{y}}$. The simplest structure is one where all the states are independent from each other. In this case a node can enter any state regardless of the state of any other node and the joint distribution simplifies to a product of simple probabilities. Such a simple structure,

[1] Labels $\boldsymbol{o}_u$ can also be discrete similar to the symbols in HMM in section 3.3.1. Extending the model for discrete symbols can be done in the same fashion.



(a) a (b) b

Fig. 3.2: Notation in tree structures (a), Example of an underlying hidden state (states in gray) process emitting labels (b).

however, does not capture the structural information in the trees induced by the parent-child relationship of the nodes. A more appropriate structure is to make each node u dependent on its parent $\rho(u)$. Thus, the state Q_u is conditioned on the state $Q_{\rho(u)}$ of its parent. Such a structure implements a first-order Markov property. Moreover, we assume that when the model labels the tree it does not reveal the states Q_u entered. Thus, the underlying process that generates a tree $\boldsymbol{y}$ is *hidden* from us, and only the labels $\boldsymbol{o}_u$, $u \in \mathcal{U}_{\boldsymbol{y}}$ can be observed. This is illustrated in Fig. 3.2(b).

The resulting model, called the *hidden Markov tree model* (HMTM) [Crouse et al., 1998, Durand and Gonçalvès, 2001], is an extension of the HMM. An HMTM models tree structure $\boldsymbol{y}$ by expressing a joint probability density for the set of hidden state variables $Q_1, \ldots, Q_{U_{\boldsymbol{y}}}$, each defined on the support $\{1, 2, \ldots, K\}$, and the set of labels $\boldsymbol{o}_1, \ldots, \boldsymbol{o}_{U_{\boldsymbol{y}}}$ in $\mathbb{R}^d$. The model is called *hidden* because the states cannot be directly observed, while *Markov* refers to the fact that the current state of a node depends only on that of its immediate predecessor (parent).

An HMTM, in the same fashion as an HMM, is defined by three sets of parameters:

- initial probability vector $\boldsymbol{\pi} = \{p(Q_1 = k)\}_{k=1,\ldots,K}$ – each element expressing the probability of the root node being in state $k \in \{1, 2, \ldots, K\}$.

- transition probability matrix $\boldsymbol{B} = \{p(Q_u = l | Q_{\rho(u)} = k)\}_{k,l=1,\ldots,K}$ – each element expressing the probability of transiting from parent $\rho(u)$ in state k to the child node u in state l. This probability is assumed to be position-invariant. Note that the sum of elements b_{kl}

over $l = 1, \ldots, K$, i.e. the sum of transitions from a state k to all other K states, must equal 1.

- the emission parameters that parametrise Gaussian distributions, $f(.; \boldsymbol{\psi}_k)$, with $\boldsymbol{\psi}_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ one for each state $k = 1, \ldots, K$. Here, $\boldsymbol{\mu}_k \in \mathbb{R}^d$ and $\boldsymbol{\Sigma}_k$ are the mean and covariance matrix, respectively, of the Gaussian associated with emission process in state k.

We shall reuse notation $\boldsymbol{\Theta}$ to refer to this collection of HMTM parameters. The Markovian dependencies of hidden states are realised by the following conditions [Durand and Gonçalvès, 2001]:

- Given the parent state $Q_{\rho(u)}$, the child state Q_u is conditionally independent of all other variables in the tree, apart from those that belong in the subtree $\boldsymbol{y}_u$:

$$p(Q_u = q_u | \{Q_v = q_v\}_{v \in \mathcal{U}_{\boldsymbol{y}}, v \notin \boldsymbol{y}_u}, \{\boldsymbol{O}_v = \boldsymbol{o}_v\}_{v \in \mathcal{U}_{\boldsymbol{y}}, v \notin \boldsymbol{y}_u}) = p(Q_u = q_u | Q_{\rho(u)} = q_{\rho(u)}). \tag{3.73}$$

- Given the (hidden) state of a node, the corresponding label is conditionally independent of all other variables in the tree:

$$p(\boldsymbol{O}_u = \boldsymbol{o}_u | \{\boldsymbol{O}_v = \boldsymbol{o}_v\}_{v \in \mathcal{U}_{\boldsymbol{y}}, v \neq u}, \{Q_v = q_v\}_{v \in \mathcal{U}_{\boldsymbol{y}}}) = p(\boldsymbol{O}_u = \boldsymbol{o}_u | Q_u = q_u). \tag{3.74}$$

Thus, the HMTM distribution can be factorised as follows:

$$\begin{aligned} p(\boldsymbol{y}, Q_1 = q_1, \ldots, Q_{U_{\boldsymbol{y}}} = q_{U_{\boldsymbol{y}}}) &= p(Q_1 = q_1) \prod_{u \in \mathcal{U}_{\boldsymbol{y}}, u \neq 1} p(Q_u = q_u | Q_{\rho(u)} = q_{\rho(u)}) \\ &\times \prod_{u \in \mathcal{U}_{\boldsymbol{y}}} p(\boldsymbol{O}_u = \boldsymbol{o}_u | Q_u = q_u). \end{aligned} \tag{3.75}$$

Henceforth, for brevity we shall drop stating both random variables and their instantiations, keeping only the latter.

Similarly to the forward-backward algorithm for HMM, the likelihood of an HMTM can be efficiently computed by the upward-downward algorithm. The motivation of this algorithm stems again from the observation that a direct calculation of likelihood without knowledge of the hidden states requires an exponential number of steps. For a tree $\boldsymbol{y}$ the upward algorithm defines the following quantity:

$$\beta_k(u; \boldsymbol{\theta}) = p(\boldsymbol{y}_u | q_u = k, \boldsymbol{\theta}). \tag{3.76}$$

Based on this, we formulate the upward-recursion [Durand et al., 2004]:

- The recursion starts from the leaves u of the tree:

$$\beta_k(u;\boldsymbol{\theta}) = p(\boldsymbol{o}_u|q_u = k, \boldsymbol{\theta}). \tag{3.77}$$

- Recursive step for non-leaf nodes u:

$$\begin{aligned}
\beta_k(u;\boldsymbol{\theta}) &= p(\boldsymbol{y}_u|q_u = k, \boldsymbol{\theta}) \\
&= \left\{ \prod_{v \in ch(u)} p(\boldsymbol{y}_v|q_u = k, \boldsymbol{\theta}) \right\} p(\boldsymbol{o}_u|q_u = k, \boldsymbol{\theta}) \\
&= \left\{ \prod_{v \in ch(u)} \sum_{i}^{K} p(\boldsymbol{y}_v|q_v = i, \boldsymbol{\theta}) p(q_v = i|q_u = k, \boldsymbol{\theta}) \right\} p(\boldsymbol{o}_u|q_u = k, \boldsymbol{\theta}) \\
&= \left\{ \prod_{v \in ch(u)} \sum_{i}^{K} \beta_i(v;\boldsymbol{\theta}) p(q_v = i|q_u = k, \boldsymbol{\theta}) \right\} p(\boldsymbol{o}_u|q_u = k, \boldsymbol{\theta}).
\end{aligned} \tag{3.78}$$

- Final step:

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_{k}^{K} \beta_k^{(n)}(1;\boldsymbol{\theta}) p(q_1 = k|\boldsymbol{\theta}). \tag{3.79}$$

Similarly, we define the downward probability:

$$\alpha_k(u;\boldsymbol{\theta}) = p(q_u = k, \boldsymbol{y}_{1\backslash u}|\boldsymbol{\theta}), \tag{3.80}$$

which is the probability of node u being at state k and observing the entire tree $\boldsymbol{y}$ apart from subtree $\boldsymbol{y}_u$. Based on this, we formulate the downward-recursion [Durand et al., 2004]:

- The recursion starts from the root u_1 of the tree:

$$\alpha_k(1;\boldsymbol{\theta}) = p(q_1 = k|\boldsymbol{\theta}). \tag{3.81}$$

- Recursive step for all nodes u in the tree, apart from root node:

$$\begin{aligned}
\alpha_k^{(n)}(u;\boldsymbol{\theta}) &= p(q_u = k, \boldsymbol{y}_{1\setminus u}, \boldsymbol{\theta}) = \sum_{i=1}^{K} p(q_u = k, q_{\rho(u)} = i, \boldsymbol{y}_{1\setminus u}, \boldsymbol{\theta}) \\
&= \sum_{i=1}^{K} p(q_u = k | q_{\rho(u)} = i, \boldsymbol{\theta}) \frac{p(\boldsymbol{y}_{\rho(u)} | q_{\rho(u)} = i, \boldsymbol{\theta}) p(q_{\rho(u)} = i, \boldsymbol{y}_{1\setminus\rho(u)}, \boldsymbol{\theta})}{\sum_{j=1}^{K} p(\boldsymbol{y}_u | q_u = j, \boldsymbol{\theta}) p(q_u = j | q_{\rho(u)} = i, \boldsymbol{\theta})} \\
&= \sum_{i=1}^{K} p(q_u = k | q_{\rho(u)} = i, \boldsymbol{\theta}) \frac{\beta_i(\rho(u);\boldsymbol{\theta}) \alpha_i(\rho(u);\boldsymbol{\theta})}{\sum_{j=1}^{K} \beta_j(u;\boldsymbol{\theta}) p(q_u = j | q_{\rho(u)} = i, \boldsymbol{\theta})}. \qquad (3.82)
\end{aligned}$$

Thus, the model likelihood of a tree $\boldsymbol{y}$ can be calculated as follows:

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \alpha_k(u;\boldsymbol{\theta}) \beta_k(u;\boldsymbol{\theta}). \qquad (3.83)$$

3.4.2 Training of Hidden Markov Tree Models

The model likelihood for a dataset of independently generated data items $\mathcal{D} = \{\boldsymbol{y}^{(1)}, ..., \boldsymbol{y}^{(N)}\}$ is:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) &= \prod_{n=1}^{N} p(\boldsymbol{y}^{(n)}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \sum_{\boldsymbol{q}\in\{1,2,...,K\}^{\mathcal{U}_{\boldsymbol{y}^{(n)}}}} p(q_1|\boldsymbol{\theta}) \prod_{u\in\mathcal{U}_{\boldsymbol{y}^{(n)}}, u\neq 1}^{U_n} p(q_u | q_{\rho(u)}, \boldsymbol{\theta}) \\
&\times \prod_{u\in\mathcal{U}_{\boldsymbol{y}^{(n)}}}^{U_n} p(\boldsymbol{o}_u^{(n)} | q_t, \boldsymbol{\theta}), \qquad (3.84)
\end{aligned}$$

where we denote the number of nodes $U_{\boldsymbol{y}^{(n)}}$ of the n-th tree $\boldsymbol{y}^{(n)}$ by U_n. We require the likelihood to be maximised. This can be achieved by adopting an EM formulation of the problem by writing the (complete data) likelihood in terms of hidden indicator variables z, collectively referred to as $\mathcal{Z}$:

$$z_k^{(n,u)} = \begin{cases} 1, & \text{if for tree } \boldsymbol{y}^{(n)} \text{ node } u \text{ was in state } k; \\ 0, & \text{otherwise.} \end{cases}$$

$$z_{k\to l}^{(n,u)} = \begin{cases} 1, & \text{if for tree } \boldsymbol{y}^{(n)} \text{ node } u \text{ was in state } l \text{ and its parent } \rho(u) \text{ was in state } k; \\ 0, & \text{otherwise.} \end{cases}$$

Using the above indicator variables and based on Eq. (3.57), we can rewrite the model likelihood and take the logarithm as follows:

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{D},\mathcal{Z}) = \prod_{n=1}^{N} p(\boldsymbol{y}^{(n)}|\boldsymbol{\theta}) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(q_1=k|\boldsymbol{\theta})^{z_k^{(n,1)}} \prod_{u\in\mathcal{U}_{\boldsymbol{y}^{(n)}},u\neq 1}^{U_n}\prod_{k=1}^{K}\prod_{l=1}^{K} p(q_u=l|q_{\rho(u)}=k,\boldsymbol{\theta})^{z_{k\to l}^{(n,u)}}$$
$$\times \prod_{u\in\mathcal{U}_{\boldsymbol{y}^{(n)}}}^{U_n}\prod_{k=1}^{K} p(\boldsymbol{o}_u^{(n)}|q_u,\boldsymbol{\theta})^{z_k^{(n,u)}}, \tag{3.85}$$

$$\log\mathcal{L}(\boldsymbol{\theta}|\mathcal{D},\mathcal{Z}) = \sum_{n=1}^{N}\Bigg(\sum_{k=1}^{K} z_k^{(n,1)}\log p(q_1=k|\boldsymbol{\theta}) + \sum_{u\in\mathcal{U}_{\boldsymbol{y}^{(n)}},u\neq 1}^{U_n}\sum_{k=1}^{K}\sum_{l=1}^{K} z_{k\to l}^{(n,u)}\log p(q_u=l|q_{\rho(u)}=k,\boldsymbol{\theta})$$
$$+ \sum_{u\in\mathcal{U}_{\boldsymbol{y}^{(n)}}}^{U_n}\sum_{k=1}^{K} z_k^{(n,u)}\log p(\boldsymbol{o}_u^{(n)}|q_u,\boldsymbol{\theta})\Bigg). \tag{3.86}$$

Following the EM formulation, we maximise instead the expected complete data log-likelihood, a lower bound of the likelihood. In the E-step, the hidden variables are estimated by their posterior expectation given the observed data and the current parameters $\boldsymbol{\theta}^{(i)}$ at the i-th iteration [Crouse et al., 1998]:

$$E[z_k^{(n,u)}|\mathcal{D},\boldsymbol{\theta}^{(i)}] = p(q_u=k|\boldsymbol{y}^{(n)},\boldsymbol{\theta}^{(i)}) = \frac{p(q_u=k,\boldsymbol{y}^{(n)}|\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{y}^{(n)}|\boldsymbol{\theta}^{(i)})} = \frac{\alpha_k(u;\boldsymbol{\theta}^{(i)})\beta_k(u;\boldsymbol{\theta}^{(i)})}{\sum_{l=1}^{K}\alpha_l(u;\boldsymbol{\theta}^{(i)})\beta_l(u;\boldsymbol{\theta}^{(i)})}, \tag{3.87}$$

$$
\begin{aligned}
E[z_{k\to l}^{(n,u)}|\mathcal{D},\boldsymbol{\theta}^{(i)}] &= p(q_u=l,q_{\rho(u)}=k|\boldsymbol{y}^{(n)},\boldsymbol{\theta}^{(i)}) = \frac{p(q_u=l,q_{\rho(u)}=k,\boldsymbol{y}^{(n)}|\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{y}^{(n)}|\boldsymbol{\theta}^{(i)})} \\
&= \frac{p(q_u=l,q_{\rho(u)}=k,\boldsymbol{y}_{1\backslash\rho(u)}^{(n)},\boldsymbol{y}_{\rho(u)\backslash u}^{(n)},\boldsymbol{y}_u^{(n)}|\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{y}^{(n)}|\boldsymbol{\theta}^{(i)})} \\
&= \frac{p(\boldsymbol{y}_u^{(n)}|q_u=l,\boldsymbol{\theta}^{(i)})p(q_u=l|q_{\rho(u)}=k|\boldsymbol{\theta}^{(i)})p(\boldsymbol{y}_{\rho(u)\backslash u}^{(n)}|q_{\rho(u)}=k,\boldsymbol{\theta}^{(i)})p(q_{\rho(u)}=k,\boldsymbol{y}_{1\backslash\rho(u)}^{(n)}|\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{y}^{(n)}|\boldsymbol{\theta}^{(i)})} \\
&= \frac{p(\boldsymbol{y}_u^{(n)}|q_u=l,\boldsymbol{\theta}^{(i)})p(q_u=l|q_{\rho(u)}=k|\boldsymbol{\theta}^{(i)})\frac{p(\boldsymbol{y}_{\rho(u)}^{(n)}|q_{\rho(u)}=k,\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{y}_u^{(n)}|q_{\rho(u)}=k,\boldsymbol{\theta}^{(i)})}p(q_{\rho(u)}=k,\boldsymbol{y}_{1\backslash\rho(u)}^{(n)}|\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{y}^{(n)}|\boldsymbol{\theta}^{(i)})} \\
&= \frac{p(\boldsymbol{y}_u^{(n)}|q_u=l,\boldsymbol{\theta}^{(i)})p(q_u=l|q_{\rho(u)}=k|\boldsymbol{\theta}^{(i)})\frac{p(\boldsymbol{y}_{\rho(u)}^{(n)}|q_{\rho(u)}=k,\boldsymbol{\theta}^{(i)})}{\sum_j p(\boldsymbol{y}_u^{(n)}|q_u=j,\boldsymbol{\theta}^{(i)})p(q_u=j|q_{\rho(u)}=k,\boldsymbol{\theta}^{(i)})}}{p(\boldsymbol{y}^{(n)}|\boldsymbol{\theta}^{(i)})} \\
&\times \frac{p(q_{\rho(u)}=k,\boldsymbol{y}_{1\backslash\rho(u)}^{(n)}|\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{y}^{(n)}|\boldsymbol{\theta}^{(i)})} \\
&= \frac{\beta_l^{(n)}(u;\boldsymbol{\theta}^{(i)})p(q_u=l|q_{\rho(u)}=k,\boldsymbol{\theta}^{(i)})\frac{\beta_k^{(n)}(\rho(u);\boldsymbol{\theta}^{(i)})}{\sum_j p(\beta_j^{(n)}(u;\boldsymbol{\theta}^{(i)})p(q_u=j|q_{\rho(u)}=k,\boldsymbol{\theta}^{(i)})}\alpha_k^{(n)}(\rho(u);\boldsymbol{\theta}^{(i)})}{\sum_{l=1}^K \alpha_l^{(n)}(u;\boldsymbol{\theta}^{(i)})\beta_l^{(n)}(u;\boldsymbol{\theta}^{(i)})},
\end{aligned}
\tag{3.88}
$$

where we have augmented notation $\alpha(u;\boldsymbol{\theta}^{(i)}), \beta(u;\boldsymbol{\theta}^{(i)})$ with index n to denote the downward and upward probabilities for the n-th tree, $\alpha^{(n)}(u;\boldsymbol{\theta}^{(i)})$ and $\beta^{(n)}(u;\boldsymbol{\theta}^{(i)})$ respectively. We express the expected complete-data log-likelihood as:

$$
\begin{aligned}
E_{\mathcal{Z}}[\log\mathcal{L}(\boldsymbol{\theta}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\theta}^{(i)}] &= \sum_{n=1}^N \Bigg(\sum_{k=1}^K p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{\theta}^{(i)})\log p(q_1|\boldsymbol{\theta}) \\
&+ \sum_{u\in\mathcal{U}_{\boldsymbol{y}^{(n)}},u\neq 1}^{U_n} \sum_{l=1}^K\sum_{k=1}^K p(q_u=l,q_{\rho(u)}=k|\boldsymbol{y}^{(n)},\boldsymbol{\theta}^{(i)})\log p(q_u=l|q_{\rho(u)}=k,\boldsymbol{\theta}) \\
&+ \sum_{u\in\mathcal{U}_{\boldsymbol{y}^{(n)}}}^{U_n}\sum_{k=1}^K p(q_u=k|\boldsymbol{y}^{(n)},\boldsymbol{\theta}^{(i)})\log p(\boldsymbol{o}_u^{(n)}|q_u=k,\boldsymbol{\theta})\Bigg).
\end{aligned}
\tag{3.89}
$$

In the M-step we calculate the derivatives of (3.89) with respect to the parameters of the model, which lead to the following update equations [Crouse et al., 1998]:

$$
\pi_k = \frac{1}{N}\sum_{n=1}^N p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{\theta}^{(i)}),
\tag{3.90}
$$

$$b_{kl} = \frac{\sum_{n=1}^{N} \sum_{u \in \mathcal{U}_{\boldsymbol{y}^{(n)}}}^{U_n} p(q_u = l, q_{\rho(u)} = k | \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(i)})}{\sum_{n=1}^{N} \sum_{u \in \mathcal{U}_{\boldsymbol{y}^{(n)}}}^{U_n} p(q_{\rho(u)} = k | \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(i)})}. \tag{3.91}$$

If the emission distribution $f(.; \boldsymbol{\psi}_k)$ assumes the form of a Gaussian, then its parameters $\boldsymbol{\psi} = \{\boldsymbol{\psi}_k\}_{k=1:K}$, with $\boldsymbol{\psi} = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ the mean and the covariance matrix for each state k respectively, are updated as follows:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} \sum_{u \in \mathcal{U}_{\boldsymbol{y}^{(n)}}}^{U_n} \boldsymbol{o}_u^{(n)} p(q_u = k | \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(i)})}{\sum_{n=1}^{N} \sum_{u \in \mathcal{U}_{\boldsymbol{y}^{(n)}}}^{U_n} p(q_u = k | \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(i)})}, \tag{3.92}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N} \sum_{u \in \mathcal{U}_{\boldsymbol{y}^{(n)}}}^{U_n} (\boldsymbol{o}_u^{(n)} - \boldsymbol{\mu}_k)^T (\boldsymbol{o}_u^{(n)} - \boldsymbol{\mu}_k) p(q_u = k | \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(i)})}{\sum_{n=1}^{N} \sum_{u \in \mathcal{U}_{\boldsymbol{y}^{(n)}}}^{U_n} p(q_u = k | \boldsymbol{y}^{(n)}, \boldsymbol{\theta}^{(i)})}. \tag{3.93}$$

3.4.3 Mixtures of Hidden Markov Tree Models

Mixtures of HMTMs are formulated in the precise same fashion as mixtures of HMMs and Gaussian densities. Thus, we shall not reiterate the mixture formulation and the respective EM optimisation machinery, especially since in section 4.2 we elaborate on a constrained mixture of HMTMs.

3.4.4 Overview of Markov Tree Models

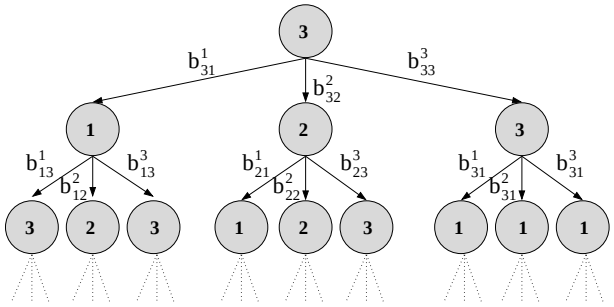


Fig. 3.3: An example of a 3-regular tree. Nodes are labelled from the set $\{1, 2, 3\}$ and edges are annotated with the transition probabilities.

This section presents an additional generative model, the *Markov Tree Model* (MTM), for density modelling of tree structures. Here we concentrate on a particular class of trees, namely R-regular trees, which is the class of trees where the outdegree of parent nodes is fixed to R, i.e. each parent node has exactly R children. A MTM is an observable process, that generates a label $\boldsymbol{o}_u = 1, \ldots, K$ for each node $u \in \mathcal{U}_{\boldsymbol{y}}$ of tree $\boldsymbol{y}$. Because it is observable, all information

of the generative process, namely the set of assigned labels $\boldsymbol{o}_u$, is directly available. This is in contrast to HMMs and HMTMs where a non-observable hidden state q_u is associated with each node u. The process labels the tree in a top-down fashion, starting from the root, node u_1, of the tree and working down to the leaves of the tree. At each transition from a parent node $\rho(u)$ to a child node $u_r, r = 1, \ldots, R$, a label $\boldsymbol{o}_{u_r}$ is assigned. The label assignment is conditionally dependent on the label $\boldsymbol{o}_{\rho(u)}$ of the parent node $\rho(u)$ and the position of the child, i.e. whether its the 1-st, 2-nd ... or R-th child. This dependency is expressed as a probability $p(\boldsymbol{o}_u|\boldsymbol{o}_{\rho(u)}, pos(u))$, where $pos(\cdot)$ is a function $pos : u \rightarrow \{1, 2, \ldots, R\}$, that returns the position of node u. A MTM is a first-order Markov process, where the label of a node is conditionally independent from all labels that belong to ancestor nodes, given its parent node and position.

The transitions are governed by R transition matrices $\boldsymbol{B}^{(r)}$, one for each child position $r = 1, \ldots, R$, with entries $b^r_{kl} = p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k)$ for $k, l = 1, \ldots K$. Given a tree $\boldsymbol{y}$ the MTM likelihood is simply:

$$p(\boldsymbol{y}) = p(\boldsymbol{o}_1) \prod_{u=2}^{U_n} p(\boldsymbol{o}_u|\boldsymbol{o}_{\rho(u)}, pos(u)). \tag{3.94}$$

Note that we impose a flat probability for the initial state probability distribution of the root node, $p(\boldsymbol{o}_1) = \frac{1}{K}$.

3.4.5 Training of Markov Tree Models

The (scaled[2]) model likelihood for a dataset $\mathcal{D} = \{\boldsymbol{y}^{(1)}, ..., \boldsymbol{y}^{(N)}\}$ of independently generated trees, is expressed as:

$$\mathcal{L}(\boldsymbol{B}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{B}) \propto \prod_{n=1}^{N} \prod_{u=2}^{U_n} p(\boldsymbol{o}_u|\boldsymbol{o}_{\rho(u)}, pos(u)). \tag{3.95}$$

It is useful to rewrite the (scaled) likelihood in (3.95) as:

$$\mathcal{L}(\boldsymbol{B}|\mathcal{D}) \quad \propto \quad \prod_{n=1}^{N} \prod_{u=2}^{U_n} \prod_{k=1}^{K} \prod_{l=1}^{K} \prod_{r=1}^{R} p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r)^{\delta_{\boldsymbol{o}_{\rho(u)},k} \delta_{\boldsymbol{o}_u,l} \delta_{pos(u),r}}, \tag{3.96}$$

[2]We discard the flat probability of root node $p(\boldsymbol{o}_1) = \frac{1}{K}$.

$$
\begin{aligned}
\log \mathcal{L}(\boldsymbol{B}|\mathcal{D}) &\propto \sum_{n=1}^{N}\sum_{u=2}^{U_n}\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{r=1}^{R}\delta_{\boldsymbol{o}_{\rho(u)},k}\delta_{\boldsymbol{o}_u,l}\delta_{pos(u),r}\log p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r) \\
&\propto \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{r=1}^{R}\left(\sum_{u=2}^{U_n}\delta_{\boldsymbol{o}_{\rho(u)},k}\delta_{\boldsymbol{o}_u,l}\delta_{pos(u),r}\right)\log p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r) \\
&\propto \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{r=1}^{R}\nu_{rkl}^{(n)}\log p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r), \qquad (3.97)
\end{aligned}
$$

where $\delta_{i,j}$ is the Kronecker delta function (with $\delta_{i,j} = 1$ for $i = j$ and $\delta_{i,j} = 0$ for $i \neq j$), $\nu_{rkl}^{(n)}$ is a count of how many times the transition from a parent node labelled by k to the r-th child labelled by l occurs in tree $\boldsymbol{y}^{(n)}$. We proceed to the optimisation of the model by calculating derivatives of the model log-likelihood with respect to parameter b_{ij}^r. In doing so, we must take into account the constraint $\sum_{j=1}^{K} b_{ij}^h = 1$, meaning that the sum of probabilities when transiting from label i to all other labels j must be equal to 1. Hence, we include the Lagrange multiplier λ:

$$
\begin{aligned}
\frac{\partial}{b_{ij}^h}\log \mathcal{L}(\boldsymbol{B}|\mathcal{D}) &\propto \frac{\partial}{\partial b_{ij}^h}\left[\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{r=1}^{R}\nu_{rkl}^{(n)}\log b_{kl}^r - \lambda\left(\sum_{j=1}^{K} b_{ij}^h - 1\right)\right] \\
&\propto \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{r=1}^{R}\nu_{rkl}^{(n)}\frac{\partial}{\partial b_{ij}^h}\log b_{kl}^r - \frac{\partial}{\partial b_{ij}^h}\lambda\left(\sum_{j=1}^{K} b_{ij}^h - 1\right) \\
&\propto \sum_{n=1}^{N}\nu_{hij}^{(n)}\frac{1}{b_{ij}^h} - \lambda. \qquad (3.98)
\end{aligned}
$$

Set (3.98) to zero and sum over all labels $j = 1, \ldots, K$:

$$
\begin{aligned}
\frac{\partial}{\partial b_{ij}^h}\log \mathcal{L}(\boldsymbol{B}|\mathcal{D}) &= 0, \\
\sum_{j=1}^{K}\left(\sum_{n=1}^{N}\nu_{hij}^{(n)}\frac{1}{b_{ij}^h} - \lambda\right) &= 0, \\
\sum_{j=1}^{K}\left(\sum_{n=1}^{N}\nu_{hij}^{(n)} - b_{ij}^h\lambda\right) &= 0, \\
\sum_{n=1}^{N}\nu_{hi}^{(n)} - \lambda &= 0, \qquad (3.99)
\end{aligned}
$$

to obtain $\lambda = \sum_{n=1}^{N}\nu_{hi}^{(n)}$. We define $\nu_{hi}^{(n)}$ as a count of labels $\boldsymbol{o}_u = i$ of all nodes u in position

$pos(u) = h$ in tree $\boldsymbol{y}^{(n)}$. Return to (3.98) and substitute λ:

$$\begin{aligned} \sum_{n=1}^{N} \nu_i^{(n)} &= \sum_{n=1}^{N} \nu_{ij}^{(n)} \frac{1}{b_{ij}^h}, \\ b_{ij}^h &= \frac{\sum_{n=1}^{N} \nu_{hij}^{(n)}}{\sum_{n=1}^{N} \nu_{hi}^{(n)}}. \end{aligned} \tag{3.100}$$

Thus, learning of parameter b_{ij}^h is performed in a single pass which simply involves counting labels.

3.4.6 Mixtures of Markov Tree Models

Once more, we note that mixtures of MTMs are formulated in precisely the same fashion as mixtures of HMMs and Gaussian densities. In section 4.5 we shall elaborate on a constrained mixture of MTMs.

3.5 Constrained Mixture Models

The generative probabilistic models presented in this chapter are suitable for density modelling. Moreover, their respective mixtures extend their modelling capabilities to datasets with multiple clusters. The generative nature of such models allows us to modify them for specialised purposes in a transparent way. One can place constraints on the model parameters as a way of incorporating domain specific knowledge or as a way of exploring the dataset under such constraints. To clarify our discussion we consider an artificial, yet concrete example of a constrained model.

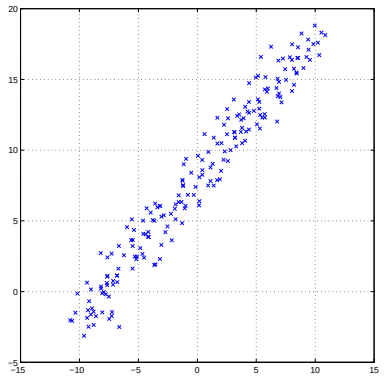


Fig. 3.4: Noisy, intrinsically one-dimensional dataset.

We wish to model a noisy, intrinsically one-dimensional dataset $\mathcal{D} \supseteq \mathbb{R}^2, \mathcal{D} = \{\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(N)}\}$.

Such a dataset is presented in Fig. 3.4. We employ a mixture of C Gaussian densities, however, this time we impose that the means of the Gaussians belong to a straight line. Imposing that all means lie on a line is straightforward; we simply require that for the two-dimensional means $\boldsymbol{\mu}_c \in \mathcal{D}$ of each component c, the second coordinate of $\boldsymbol{\mu}_c$ is generated by the line equation:

$$\boldsymbol{\mu}_c = [x_c,\ (\alpha x_c + \beta)]^T, \tag{3.101}$$

where α and β are the slope and intercept of the line and $x_c \in \mathbb{R}$. Thus, the two-dimensional means $\boldsymbol{\mu}_c$ have only one degree of freedom. We define function $\boldsymbol{l}(x_c) = [x_c,\ (\alpha x_c + \beta)]^T$. The model likelihood given the dataset is:

$$\mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) = \prod_{n=1}^{N} \sum_{c=1}^{C} P(c) p(\boldsymbol{t}^{(n)}|c). \tag{3.102}$$

where $p(\boldsymbol{t}|c) = \mathcal{N}(\boldsymbol{t}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma}_c)$ and $\boldsymbol{\Theta} = \left\{ \alpha, \beta, \{\boldsymbol{\Sigma}_c\}_{c=1\ldots C}, \{x_c\}_{c=1\ldots C} \right\}$ are the free parameters of the model. For ease of exposition we make two simplifications; one is setting a common variance $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}$ and the second is fixing the priors to be equal to each other, $P(c) = \frac{1}{C}$.

Following standard EM methodology presented in sections 3.2 and 3.2.1, we maximise instead the (scaled[3]) expected complete-data log-likelihood that acts as a lower bound to the log-likelihood of the model:

$$E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\Theta}^{(i)}] \quad \propto \quad \sum_{n=1}^{N} \sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \log p(\boldsymbol{t}^{(n)}|c). \tag{3.103}$$

where hidden variables z on the component origin of data items are introduced and posteriors $p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta})$ are calculated in the same way as in section 3.2. We proceed to the M-step by taking derivatives of (3.103) with respect to the parameters in $\boldsymbol{\Theta}$ and set them to zero. For convenience we define $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(i)}) = E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\Theta}^{(i)}]$ (more details on the derivations can be found

[3]Scaled since the equal and fixed priors $P(c) = \frac{1}{C}$ are discarded.

in appendix A):

$$
\begin{aligned}
&\frac{\partial}{\partial x_c} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(i)}) = 0, \\
&\frac{\partial}{\partial x_c} \sum_{n=1}^{N} \sum_{c'=1}^{C} p(c'|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \log p(\boldsymbol{t}^{(n)}|c') = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{\partial}{\partial x_c} \log p(\boldsymbol{t}^{(n)}|c) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{\partial}{\partial x_c} \log \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma}) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{1}{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma})} \frac{\partial}{\partial x_c} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma}) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{1}{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma})} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma}) \frac{\partial}{\partial x_c} (-\frac{1}{2} (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (-(\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c)))^T \boldsymbol{\Sigma}^{-1} \frac{\partial}{\partial x_c} (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c)) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (t_1^n - x_c) = 0, \\
&x_c = \frac{\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) t_1^{(n)}}{\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})}.
\end{aligned}
\tag{3.104}
$$

where $t_1^{(n)}$ is the first coordinate of $\boldsymbol{t}^{(n)} = [t_1^{(n)},\ t_2^{(n)}]^T$.

Similarly, we determine α, β and $\boldsymbol{\Sigma}$:

$$
\begin{aligned}
&\frac{\partial}{\partial\alpha}Q(\boldsymbol{\Theta},\boldsymbol{\Theta}^{(i)})=0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})\frac{\partial}{\partial\alpha}\log p(\boldsymbol{t}^{(n)}|c)=0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})\frac{\partial}{\partial\alpha}\log\mathcal{N}(\boldsymbol{t}^{(n)};\boldsymbol{l}(x_c),\boldsymbol{\Sigma})=0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})\frac{1}{\mathcal{N}(\boldsymbol{t}^{(n)};\boldsymbol{l}(x_c),\boldsymbol{\Sigma})}\frac{\partial}{\partial\alpha}\mathcal{N}(\boldsymbol{t}^{(n)};\boldsymbol{l}(x_c),\boldsymbol{\Sigma})=0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})\frac{1}{\mathcal{N}(\boldsymbol{t}^{(n)};\boldsymbol{l}(x_c),\boldsymbol{\Sigma})}\mathcal{N}(\boldsymbol{t}^{(n)};\boldsymbol{l}(x_c),\boldsymbol{\Sigma})\frac{\partial}{\partial\alpha}(-\frac{1}{2}(\boldsymbol{t}^{(n)}-\boldsymbol{l}(x_c))^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{t}^{(n)}-\boldsymbol{l}(x_c)))=0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})(-(\boldsymbol{t}^{(n)}-\boldsymbol{l}(x_c)))^T\boldsymbol{\Sigma}^{-1}\frac{\partial}{\partial\alpha}(\boldsymbol{t}^{(n)}-\boldsymbol{l}(x_c))=0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})(t_2^{(n)}-\alpha x_c-\beta)x_c=0,\\
&\alpha=\frac{\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})(t_2^{(n)}-\beta)x_c}{\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})x_c^2}.
\end{aligned}
\tag{3.105}
$$

The update equation for parameter β is derived in the same way:

$$
\begin{aligned}
&\frac{\partial}{\partial\beta}Q(\boldsymbol{\Theta},\boldsymbol{\Theta}^{(i)})=0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})\frac{\partial}{\partial\beta}\log p(\boldsymbol{t}^{(n)}|c)=0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})\frac{\partial}{\partial\beta}(t_2^{(n)}-\alpha x_c-\beta)=0,\\
&\beta=\frac{\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})(t_2^{(n)}-\alpha x_c)}{\sum_{n=1}^{N}\sum_{c=1}^{C}p(c|\boldsymbol{t}^{(n)},\boldsymbol{\Theta}^{(i)})}
\end{aligned}
\tag{3.106}
$$

As for the common covariance matrix $\boldsymbol{\Sigma}$ the same update equation (3.39) as for the mixture of Gaussians is used.

We initialise the model with random parameters from a uniform distribution and set an initial wide covariance matrix and use scaled conjugate gradients as the optimisation procedure in the M-step employing gradients (3.104)-(3.106). After a few iterations of the EM algorithm, the model discovers the clusters as shown in Fig. 3.5. Despite the trivial nature of this example,

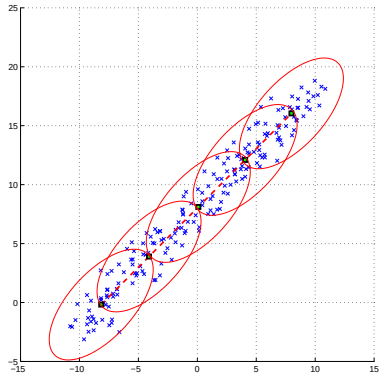


Fig. 3.5: Fitted dataset. The means of the mixture of Gaussians belong to a noisy one-dimensional line.

it clearly demonstrates how a constraint can be incorporated in the mixture of Gaussians. The constraint introduced has also a topological aspect; neighbouring means, address neighbouring Gaussian components which in turn model neighbouring data items. That is, the closer two Gaussians are, the closer the resemblance of the data items they model.

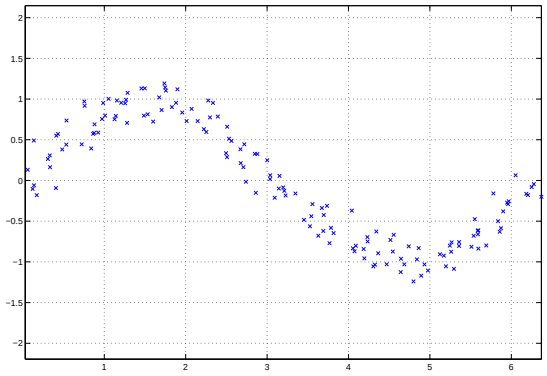


Fig. 3.6: Noisy, non-linear, intrinsically one-dimensional dataset.

To further develop this topological aspect induced by the constraint mixture model, we apply the same idea to a another dataset $\mathcal{D} \supseteq \mathbb{R}^2, \mathcal{D} = \{\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(N)}\}$, illustrated in Fig. 3.6. This $\mathcal{D}$ does not belong specifically to a "noisy" straight line, but to a general "noisy" curve. The goal is to capture its intrinsic dimensionality. One way of going about, is to embed a one-dimensional line $\ell = \{x \in [-1, +1]\}$ into the higher-dimensional data space. However, instead of embedding the entire line we discretise it into C regularly spaced points x_c and work only on these (see Fig. 3.7). Picking regularly-spaced points ensures a uniform representation of the line in the embedded space, and of course the more points we pick the better the embedding.

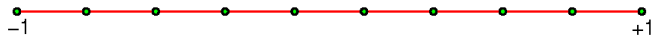


Fig. 3.7: Regularly spaced points x_c on line ℓ

The embedding is realised by a non-linear, smooth mapping Γ that maps each $x_c \in \ell$ to a point $\Gamma(x_c)$ in the data space. In order to capture the noisy nature of the data, we add some independently generated Gaussian noise to projections $\Gamma(x_c)$ which induces Gaussian densities in the data space of the form $\mathcal{N}(\cdot; \Gamma(x_c), \boldsymbol{\Sigma}_c)$ of covariance $\boldsymbol{\Sigma}_c$, where projections $\Gamma(x_c)$ act as means. We also set the covariance matrices to fixed spherical Gaussians, $\boldsymbol{\Sigma}_c = \sigma^2 I$. Mapping Γ is realised as a RBF network $\Gamma(x) = \boldsymbol{W}\boldsymbol{\phi}(x)$ with $\boldsymbol{W} \in \mathbb{R}^{M \times C}$, where M is the number of basis functions in the RBF network. RBF networks are suitable as they are universal function approximators ([Park and Sandberg, 1991]). In this model, $\boldsymbol{W}$ contains the free parameters. The parameters of the model contained in $\boldsymbol{\Theta}$ are now a function of $\boldsymbol{W}$, $\boldsymbol{\Theta}(\boldsymbol{W})$. However, for simplicity instead of $\boldsymbol{\Theta}(\boldsymbol{W})$, we write $\boldsymbol{W}$. The likelihood of the constrained mixture of Gaussians reads:

$$\mathcal{L}(\boldsymbol{W}|\mathcal{D}) \quad = \quad \prod_{n=1}^{N} \sum_{c=1}^{C} P(c) p(\boldsymbol{t}^{(n)}|c) = \prod_{n=1}^{N} \sum_{c=1}^{C} P(c) \mathcal{N}(\boldsymbol{t}^{(n)}; \Gamma(x_c), \sigma^2), \tag{3.107}$$

where we use equal, fixed priors $P(c) = \frac{1}{C}$ that can be discarded.

The free parameters to train are the elements of matrix $\boldsymbol{W}$. Again by turning the problem into a hidden variable problem we employ EM and maximise the (scaled) expected complete-data log-likelihood of the model:

$$Q(\boldsymbol{W}, \boldsymbol{W}^{(i)}) \quad = \quad \sum_{n=1}^{N} \sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{W}^{(i)}) \log \mathcal{N}(\boldsymbol{t}^{(n)}; \Gamma(x_c), \sigma^2). \tag{3.108}$$

The derivatives of $Q(\boldsymbol{W}; \boldsymbol{W}^{(i)})$ with respect to the elements w_{ij} of matrix $\boldsymbol{W}$ are obtained as

follows:

$$\begin{aligned}
\frac{\partial}{\partial w_{mj}} Q(\boldsymbol{W}, \boldsymbol{W}^{(i)}) &= \frac{\partial}{\partial w_{mj}} \sum_{n=1}^{N} \sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{W}^{(i)}) \log[p(c)\mathcal{N}(\boldsymbol{t}^{(n)}; \Gamma(x_c), \sigma^2)] \\
&= \sum_{n=1}^{N} \sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{W}^{(i)}) \frac{\partial}{\partial w_{mj}} \log[p(c)\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{W}\boldsymbol{\phi}(x_c), \sigma^2)] \\
&= \sum_{n=1}^{N} \sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{W}^{(i)}) \frac{\partial}{\partial w_{mj}} (-\frac{1}{2\sigma^2} (\boldsymbol{y}^{(n)} - \boldsymbol{W}\boldsymbol{\phi}(x_c))^T (\boldsymbol{y}^{(n)} - \boldsymbol{W}\boldsymbol{\phi}(x_c))) \\
&= \sum_{n=1}^{N} \sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{W}^{(i)}) \frac{1}{\sigma^2} (\boldsymbol{y}^{(n)} - \boldsymbol{W}\boldsymbol{\phi}(x_c))^T \boldsymbol{E}^{mj} \boldsymbol{\phi}(x_c), \qquad (3.109)
\end{aligned}$$

where $\boldsymbol{E}^{mj}$ is a matrix with all entries equal to zero apart from element m, j that is equal to unity.

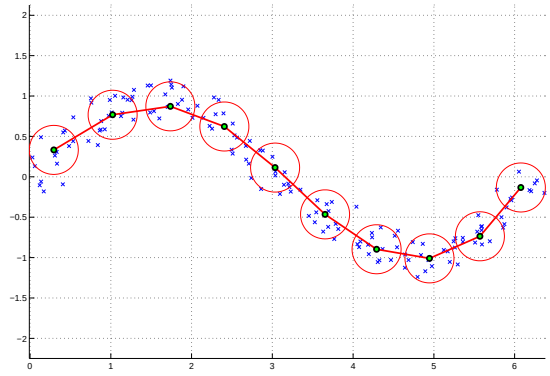


Fig. 3.8: Fitted dataset: The means of the mixture of Gaussians belong to the one-dimensional line ℓ in Fig. 3.7.

We set $C = 10$, initiate training by sampling $\boldsymbol{W}$ from a uniform distribution and employ gradient descend as our optimisation procedure for the M-step using the gradient in (3.109). After some iterations, weight matrix $\boldsymbol{W}$ is appropriately adjusted so that the constrained model adequately models the data (see Fig. 3.8). Mapping Γ maps point x_c on ℓ to a mean $\Gamma(x_c)$ in the data space, so that the Gaussians form a constrained mixture on ℓ that "explains" the distribution of data. Increasing C improves the embedding of ℓ, however, at a higher computational cost. Nevertheless, Γ does not map only the points x_c on ℓ into the data space; it actually maps the entire set of points of ℓ into the data space in a continuous way. As we traverse ℓ from -1 to $+1$ (Fig. 3.7), each point $x \in \ell$ we visit addresses (via Γ) a Gaussian that explains a certain subset of data points. Neighbouring points on ℓ, address neighbouring Gaussians and neighbouring Gaussians explain similar data points. By similarity, we refer to the closeness

of coordinates of data points, that is the closeness of data points in the data space. Thus, a one-dimensional topology is induced in the data space; neighbouring points are explained by Gaussians that are neighbours on ℓ.

This idea of embedding a lower dimensional space to a higher data space via a constrained mixture, is the fundamental idea of the *Generative Topographic Mapping* algorithm in [Svensén, 1998, Bishop et al., 1998, Bishop et al., 1996]. A constraint mixture of Gaussians is formulated with the purpose of obtaining a low-dimensional projection of the data residing in the higher dimensional space in order to visualise them. For this purpose the dimensionality of the lower space is set to two (i.e. dimension of computer screen). The algorithm is reviewed in the next chapter.

Chapter 4

The Generative Topographic Mapping Algorithm and Extensions

This chapter commences with reviewing the *Generative Topographic Mapping* (GTM) [Svensén, 1998, Bishop et al., 1998], the foundation of this work. The GTM is a probabilistic principled approach to visualising high-dimensional data based on the concept of a constrained mixture introduced in section 3.5. It constitutes an alternative approach to SOM, addressing some of the limitations that stem from its heuristic nature [Bishop et al., 1996]. In section 4.2 we present our own contribution in extending the GTM to the visualisation of tree-structured data. This extension, the *GTM-HMTM*, relies on the formulation of a constrained mixture of HMTM components. The GTM-HMTM is tested on three datasets, a toy dataset, a set of artificial images that represent houses, ships and traffic policemen expressed as trees and a set of real images expressed as quadtrees. We compare GTM-HMTM with a candidate member of the recursive neural-based approaches discussed in section 2.3, the SOMSD, and discuss the benefits brought by a probabilistic model-based formulation. Finally in section 4.5 we present another contribution, namely an alternative extension of GTM for tree-structured data that employs MTMs as local noise models, the *GTM-MTM*, accompanied by a set of experiments.

4.1 The Original Generative Topographic Mapping Algorithm

Let us consider a dataset of static d-dimensional vectors $\mathcal{D} = \left\{\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)} \ldots, \boldsymbol{t}^{(N)}\right\}$ that are independently distributed. We model the density of $\mathcal{D}$ with a mixture of C spherical Gaussians:

$$p(\mathcal{D}) = \prod_{n=1}^{N} \sum_{c=1}^{M} P(c) p(\boldsymbol{t}^{(n)}|c) = \prod_{n=1}^{N} \sum_{c=1}^{M} P(c) \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_c, \sigma_c), \tag{4.1}$$

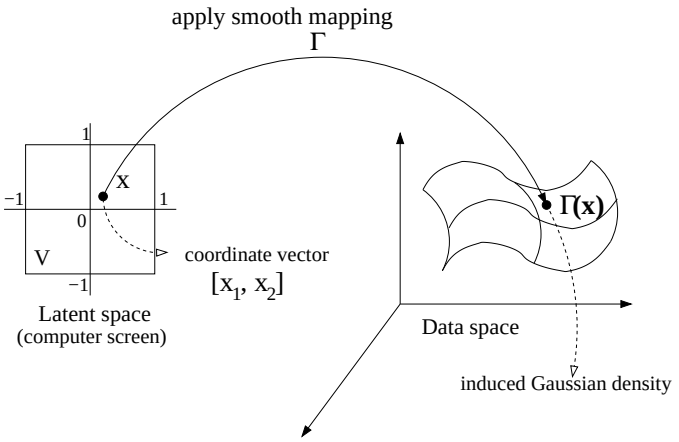


Fig. 4.1: Mapping from latent points to the means of Gaussian densities in the data space. Adapted from [Bishop et al., 1998]

where $P(c)$ are the mixing coefficients with $0 \leq P(c) \leq 1$ and $\sum_{c=1}^{C} P(c) = 1$, $\boldsymbol{\mu}_c$ the means of the Gaussians and σ_c^2 the variances. For brevity of presentation we shall assume that $P(c) = \frac{1}{C}$ and that the variance $\sigma_c^2 = \sigma^2$ is fixed. This model is the standard mixture model presented in section 3.1.2. It is an unconstrained model in the sense that its parameters, the means, do not adhere to any constraints and can move freely. This model which is useful for density modelling can be further extended to capture topographic organisation of vectorial data.

Topographic organisation can be introduced by requiring that the means of the mixture model reside on an image, under a smooth map Γ, of a continuous Euclidean latent space $\mathcal{V} = [-1, +1]^q$ of dimension $q < d$ ($q = 2$ for the purposes of visualisation). The non-linear smooth mapping $\Gamma : \mathcal{V} \rightarrow \mathbb{R}^d$ takes the form [Bishop et al., 1998]:

$$\Gamma(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x}), \tag{4.2}$$

which can be viewed as a RBF network with M radial-basis functions $\boldsymbol{\phi}(\cdot) = [\phi_1(\cdot) \ldots \phi_M(\cdot)]^T$ and weight matrix $\boldsymbol{W} \in \mathbb{R}^{D \times M}$. Matrix $\boldsymbol{W}$ contains the free parameters of the model and plays the same role as the parameter vector $\boldsymbol{\Theta}$ of mixture models, that was used throughout chapter 3. Function Γ maps each latent point $\boldsymbol{x} \in \mathcal{V}$ to a mean $\boldsymbol{\mu}$ of the model in a non-linear manner. Since Γ is smooth, the projected points will retain their local neighbourhood in the higher dimensional data space $\mathbb{R}^d$. Thus, neighbouring points in $\mathcal{V}$ will be projected to similar means in $\mathbb{R}^d$. Mapping Γ is illustrated in Fig. 4.1. We can now formulate GTM as a mixture of

Gaussians constrained on Γ-images of latent points $\boldsymbol{x} \in \mathcal{V}$. The likelihood function reads:

$$\mathcal{L}(\boldsymbol{W}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{W}) = \prod_{n=1}^{N} \int_{\boldsymbol{x}\in\mathcal{V}} P(\boldsymbol{x}) p(\boldsymbol{t}^{(n)}|\boldsymbol{x}, \boldsymbol{W}) d\boldsymbol{x}, \tag{4.3}$$

For tractability reasons we discretize the space $\mathcal{V}$ by a rectangular grid of points $\boldsymbol{x}_c$, $c = 1, \ldots, C$. This is achieved by imposing a prior distribution on the latent space:

$$P(\boldsymbol{x}) = \frac{1}{C} \sum_{c=1}^{C} \delta(\boldsymbol{x}_c - \boldsymbol{x}), \tag{4.4}$$

where $\delta(\boldsymbol{x})$ denotes the Dirac delta function, which is $\delta(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \neq 0$ and $\delta(\boldsymbol{x}) = \infty$ for $\boldsymbol{x} = 0$. Discarding the equal priors $P(c) = \frac{1}{C}$ and using (4.4), the integral in (4.3) transforms to a sum, and we rewrite the (scaled[1]) likelihood function as:

$$\begin{aligned} \mathcal{L}(\boldsymbol{W}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{W}) &\propto \prod_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{t}^{(n)}|\boldsymbol{x}_c, \boldsymbol{W}) \propto \prod_{n=1}^{N} \sum_{c=1}^{C} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x}_c), \sigma) \\ &\propto \prod_{n=1}^{N} \sum_{c=1}^{C} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_c, \sigma). \end{aligned} \tag{4.5}$$

We seek to optimise the model likelihood $\mathcal{L}(\boldsymbol{W}|\mathcal{D})$ by adjusting parameter matrix $\boldsymbol{W}$. Training the GTM proceeds by maximising the model likelihood $\mathcal{L}$ via the Expectation-Maximisation (EM) algorithm [Bishop et al., 1998] that we reviewed in section 3.2. To that end, we postulate the following hidden indicator variables on the component origin of the data items:

$$z_c^{(n)} = \begin{cases} 1, & \text{if for } \boldsymbol{t}^{(n)} \text{ was generated by component } c; \\ 0, & \text{otherwise.} \end{cases}$$

Using these variables we write the (scaled) complete-data likelihood and take the logarithm as follows:

$$\mathcal{L}(\boldsymbol{W}|\mathcal{D}, \mathcal{Z}) \propto \prod_{n=1}^{N} \prod_{c=1}^{C} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_c, \sigma)^{z_c^{(n)}}, \tag{4.6}$$

$$\log \mathcal{L}(\boldsymbol{W}|\mathcal{D}, \mathcal{Z}) \propto \sum_{n=1}^{N} \sum_{c=1}^{C} z_c^{(n)} \log \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_c, \sigma). \tag{4.7}$$

In the E-step, at the i-th iteration, the hidden indicator variables $\boldsymbol{z}$ are estimated by their

[1]Scaled since the equal and fixed priors $P(c) = \frac{1}{C}$ are discarded.

expectation given the data and the current parameters $\boldsymbol{W}^{(i)}$:

$$E[z_c^{(n)}|\mathcal{D}, \boldsymbol{W}^{(i)}] \quad = \quad p(\boldsymbol{x}_c|\boldsymbol{t}^{(n)}, \boldsymbol{W}^{(i)}) = \frac{p(\boldsymbol{t}^{(n)}|\boldsymbol{x}_c, \boldsymbol{W}^{(i)})}{\sum_{c'=1}^{C} p(\boldsymbol{t}^{(n)}|\boldsymbol{x}_{c'}, \boldsymbol{W}^{(i)})} = \frac{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_c, \sigma)}{\sum_{c'=1}^{C} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_{c'}, \sigma)}. \tag{4.8}$$

We are now ready to write the (scaled) expected complete-data log-likelihood:

$$E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{W}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{W}^{(i)}] \quad \propto \quad \sum_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{t}^{(n)}, \boldsymbol{W}^{(i)}) \log \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_c, \sigma). \tag{4.9}$$

We now take derivatives of (4.9) with respect to the elements of matrix $\boldsymbol{W}$. However in [Bishop et al., 1998] a more elegant approach is taken by first rewriting (4.9) using (4.2) and maximising it by setting its derivatives to zero:

$$\sum_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{t}^{(n)}, \boldsymbol{W}^{(i)})(\boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x}_c) - \boldsymbol{t}^{(n)})\boldsymbol{\phi}^T(\boldsymbol{x}_c) \quad = \quad 0. \tag{4.10}$$

and then expressing it in matrix form:

$$\boldsymbol{\Phi}^T \boldsymbol{G} \boldsymbol{\Phi} \boldsymbol{W} = \boldsymbol{\Phi}^T \boldsymbol{R} \boldsymbol{T}, \tag{4.11}$$

where

- Φ is a $C \times M$ matrix with element (i, c) equal to $\phi_i(\boldsymbol{x}_c)$,
- $\boldsymbol{T}$ is a $N \times D$ matrix with element (n, k) equal to $t_k^{(n)}$,
- $\boldsymbol{R}$ is a $C \times N$ matrix with element (c, n) equal to $p(\boldsymbol{x}_c|\boldsymbol{t}^{(n)}, \boldsymbol{W}^{(i)})$,
- $\boldsymbol{G}$ is a $C \times C$ diagonal matrix with element (c, c) equal to $\sum_{n=1}^{N} p(\boldsymbol{x}_c|\boldsymbol{t}^{(n)}, \boldsymbol{W}^{(i)})$.

Matrix $\boldsymbol{W}$ may now be adjusted using matrix inversion techniques. Having trained the model, it can be used for visualising the data. To that end, we note that the probability of observing a data point $\boldsymbol{t}^{(n)}$ given a latent point $\boldsymbol{x}$ is:

$$p(\boldsymbol{t}^{(n)}|\boldsymbol{x}, \boldsymbol{W}) = \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_{\boldsymbol{x}}, \sigma). \tag{4.12}$$

We can reverse this probability, using Bayes' theorem, to obtain the posterior probability of

the latent point $\boldsymbol{x}$ given $\boldsymbol{t}^{(n)}$:

$$\begin{aligned} p(\boldsymbol{x}_c|\boldsymbol{t}^{(n)}, \boldsymbol{W}) &= \frac{p(\boldsymbol{t}^{(n)}|\boldsymbol{x}, \boldsymbol{W})P(\boldsymbol{x})}{P(\boldsymbol{t}^{(n)})} = \frac{p(\boldsymbol{t}^{(n)}|\boldsymbol{x}, \boldsymbol{W})P(\boldsymbol{x})}{\sum_{c'=1}^{C} p(\boldsymbol{t}^{(n)}|\boldsymbol{x}_{c'}, \boldsymbol{W})P(\boldsymbol{x}_{c'})} \\ &= \frac{p(\boldsymbol{t}^{(n)}|\boldsymbol{x}, \boldsymbol{W})}{\sum_{c'=1}^{C} p(\boldsymbol{t}^{(n)}|\boldsymbol{x}_{c'}, \boldsymbol{W})} = \frac{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_{\boldsymbol{x}}, \sigma)}{\sum_{c'=1}^{C} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_{c'}, \sigma)}. \end{aligned} \tag{4.13}$$

We represent each data point $\boldsymbol{t}^{(n)}$ with a point $proj(\boldsymbol{t}^{(n)})$ in the latent space given by the expectation of the posterior distribution over all latent points $\boldsymbol{x}_c$:

$$proj(\boldsymbol{t}^{(n)}) = \sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{t}^{(n)}, \boldsymbol{W})\boldsymbol{x}_c. \tag{4.14}$$

4.2 Hidden Markov Tree Models as Noise Models for GTM

4.2.1 Model Formulation

This section presents an extension of GTM from vectorial to tree structured data, the GTM-HMTM. Analogously to GTM, we want to construct a grid of latent points $\boldsymbol{x}$ in a latent space $\mathcal{V}$. Each latent point $\boldsymbol{x}$ is mapped to an HMTM via a smooth non-linear mapping Γ. Since the neighbourhood of Γ-images of $\boldsymbol{x}$ is preserved, the resulting HMTMs will be topographically organised. Here the observations are no longer fixed-length vectors $\boldsymbol{t}$, but trees $\boldsymbol{y}$ as described in section 3.4.1. For each latent point $\boldsymbol{x} \in \mathcal{V}$ we calculate the likelihood $p(\boldsymbol{y}|\boldsymbol{x})$. Each data item $\boldsymbol{y}$ is subsequently mapped to the location of the map where the $p(\boldsymbol{y}|\boldsymbol{x})$ of the local latent points $\boldsymbol{x}$ is expected to be high.

We commence the formulation of the model in the spirit of [Bishop et al., 1998] and define the latent space to be $\mathcal{V} = [-1, +1]^2$. The non-linear mapping Γ is realised by an RBF network as considered previously in GTM:

$$\Gamma(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x}).$$

In this setting, however, mapping Γ maps each point $\boldsymbol{x} \in \mathcal{V}$ to a set of HMTM parameters $\{\pi(\boldsymbol{x}), \boldsymbol{B}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{x})\}$ (see section 3.4.1) of the same form, i.e. all HMTMs have the same number of states, all have discrete emissions with the same number of symbols or all are continuous with emissions of the same dimensionality (here we shall work with continuous emissions). Assuming K number of states, emissions of dimension d that are modelled by a single Gaussian, we need: K parameters for $\pi(\boldsymbol{x})$, $K \times K$ parameters for $\boldsymbol{B}(\boldsymbol{x})$ and $d \times K$ parameters for the means in $\boldsymbol{\psi}(\boldsymbol{x})$. This is a total of $K(1+K+d)$ number of parameters (note that we have not accounted for the parameters of a covariance matrix, we will elaborate on this point later). In order to obtain

the HMTM parameters, we require $\boldsymbol{W}$ to be a $K(1+K+d)\times M$ matrix, where M is the number of radial basis function in $\boldsymbol{\phi}(\cdot)$ as defined in section 4.1. Thus, the product $\boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x})$ produces a $K(1+K+d)\times 1$ vector that summarises all HMTM parameters. However, an equivalent and more convenient formulation is to define three separate matrices for initial, transition and mean parameters. Each such matrix produces a vector of parameters. Thus, we define:

- One $K \times M$ matrix $\boldsymbol{W}^{(\boldsymbol{\pi})}$ for the initial probabilities $\pi(\boldsymbol{x})$.
- One $K \times K$ matrix $\boldsymbol{W}^{(\boldsymbol{B}_k)}$ for each state $k = 1, \ldots, K$, for the transition probabilities $\boldsymbol{B}(\boldsymbol{x})$. Each matrix $\boldsymbol{W}^{(\boldsymbol{B}_k)}$ generates a vector of probabilities for transits from state k to the other K states. When these K column vectors are put in a matrix they form transition matrix $\boldsymbol{B}(\boldsymbol{x})$.
- One $d\times K$ matrix $\boldsymbol{W}^{(\boldsymbol{\psi}_k)}$ for each state $k = 1, \ldots, K$, for the means in $\boldsymbol{\psi}(\boldsymbol{x})$. Each matrix $\boldsymbol{W}^{(\boldsymbol{\psi}_k)}$ produces the means corresponding to state k.

The above matrices produce the HMTM parameters by the following RBF mappings:

$$\boldsymbol{\pi}(\boldsymbol{x}) = \{p(q_1 = k|\boldsymbol{x}_c)\}_{k=1,\ldots,K} = \boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}), \tag{4.15}$$

$$\boldsymbol{B}(\boldsymbol{x}) = \{p(q_t = l|q_{t-1} = k, \boldsymbol{x}_c)\}_{k,l=1,\ldots,K} = \{\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x})\}_{k=1,\ldots,K}, \tag{4.16}$$

$$\boldsymbol{\psi}(\boldsymbol{x}) = \{\boldsymbol{\mu}_k\}_{k=1\ldots K} = \{\boldsymbol{W}^{(\boldsymbol{\psi}_k)}\boldsymbol{\phi}(\boldsymbol{x})\}_{k=1,\ldots,K}. \tag{4.17}$$

We must pay attention to the fact that outputs of the above RBF mappings are unbounded which can result to invalid initial and transition parameters, since these parameters are probabilities inherently restricted to the range $[0, 1]$. Moreover, the sum of transition probabilities from a state k to all other K states must equal 1. Therefore, the particular mappings are not appropriate and are treated with softmax function g:

$$\boldsymbol{\pi}(\boldsymbol{x}) = \{g_k(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\}_{k=1,\ldots,K}, \tag{4.18}$$

$$\boldsymbol{B}(\boldsymbol{x}) = \{g_l(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\}_{k,l=1,\ldots K}, \tag{4.19}$$

where the softmax function is defined as $g_k((\alpha_1, \alpha_2, \cdots, \alpha_q)^T) = \frac{\exp(\alpha_k)}{\sum_{i=1}^{q}\exp(\alpha_i)}$ and restricts the outputs in the interval $[0, 1]$. Thus, we obtain the three mappings of (4.17), (4.18) and (4.19) for the emission, initial and transition probabilities respectively, that are equivalent to a single mapping Γ like the one encountered in section 4.1 for the GTM.

We assume a dataset of given trees $\mathcal{D} = \{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, ..., \boldsymbol{y}^{(N)}\}$ that are independently generated. Matrices $\boldsymbol{W}^{(\boldsymbol{\pi})}, \{\boldsymbol{W}^{(\boldsymbol{B}_k)}\}_{k=1,\ldots,K}, \{\boldsymbol{W}^{(\boldsymbol{\psi})}\}_{k=1,\ldots,K}$ are summarised by block matrix $\boldsymbol{\mathcal{W}}$.

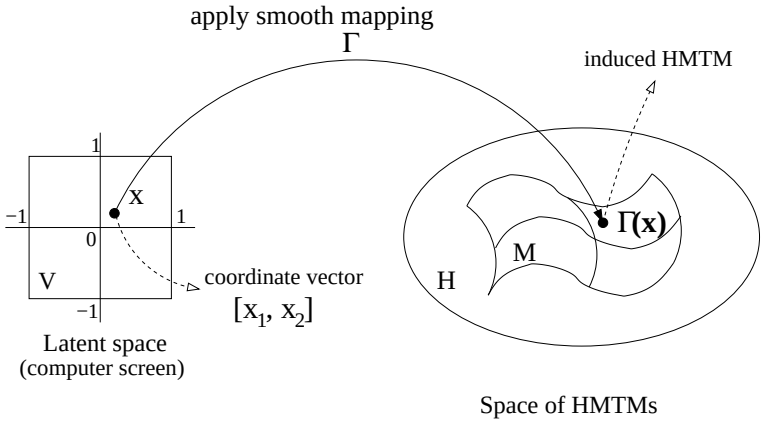


Fig. 4.2: Mapping Γ from latent space $\mathcal{V}$ to the two-dimensional manifold $\mathcal{M}$ of HMTMs.

The HMTMs addressed by the latent points $\boldsymbol{x} \in \mathcal{V}$ compose a constrained mixture of HMTMs:

$$\mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{\mathcal{W}}) = \prod_{n=1}^{N} p(\boldsymbol{y}^{(n)}|\boldsymbol{\mathcal{W}}) = \prod_{n=1}^{N} \int_{\boldsymbol{x}\in\mathcal{V}} P(\boldsymbol{x}) p(\boldsymbol{y}^{(n)}|\boldsymbol{x}, \boldsymbol{\mathcal{W}}) d\boldsymbol{x}. \tag{4.20}$$

For tractability reasons we discretize latent space $\mathcal{V}$ into a rectangular grid of points $\boldsymbol{x}_c, c = 1, \ldots, C$. This effectively imposes a prior distribution of "impulses" δ on the latent space at points $\boldsymbol{x}_c$:

$$P(\boldsymbol{x}) = \frac{1}{C} \sum_{c=1}^{C} \delta(\boldsymbol{x}_c - \boldsymbol{x}), \tag{4.21}$$

Taking into account (4.21), the likelihood in (4.20) now reads:

$$\mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D}) = \prod_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{y}^{(n)}|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) P(\boldsymbol{x}_c), \tag{4.22}$$

$$\log \mathcal{L}(\boldsymbol{W}|\mathcal{D}) \propto \sum_{n=1}^{N} \log \sum_{c=1}^{C} p(\boldsymbol{y}^{(n)}|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}}), \tag{4.23}$$

where the mixing coefficients can be ignored as $P(\boldsymbol{x}) = \frac{1}{C}$. Finally, we expand the noise models $p(\boldsymbol{y}^{(n)}|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}})$ in (4.23) using (3.75). The (scaled) log-likelihood reads:

$$\begin{aligned} \log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D}) \propto & \sum_{n=1}^{N} \log \sum_{c=1}^{C} \sum_{\boldsymbol{q}\in\{1,2,\ldots,K\}^{U_n}} p(q_1|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \prod_{u=2}^{U_n} p(q_u|q_{\rho(u)}, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \\ & \times \prod_{u=1}^{U_n} P(\boldsymbol{o}_u^{(n)}|q_u, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}). \end{aligned} \tag{4.24}$$

The formulation of the constrained model is summarised pictorially in Fig. 4.2. We want each point $\boldsymbol{x}$ in latent space $\mathcal{V}$ to address an HMTM. A smooth mapping Γ is formulated as a RBF network, that takes each point $\boldsymbol{x}$ to a point $\Gamma(\boldsymbol{x})$. Point $\Gamma(\boldsymbol{x})$ is a set of HMTM parameters and as such it addresses an HMTM. Space $\mathcal{H}$ is the set of all possible HMTM parameters that address all possible HMTMs of the same form as that imposed by (4.17),(4.18) and (4.19). However, points $\Gamma(\boldsymbol{x})$ are constrained to a two-dimensional manifold $\mathcal{M}$ that is a subspace of $\mathcal{H}$. Thus, latent space $\mathcal{V}$ is embedded into space $\mathcal{H}$ as the constrained two-dimensional manifold $\mathcal{M}$, induced via the RBF mappings. Training can be visualised as the folding and stretching of manifold $\mathcal{M}$ in response to the adjustment of parameters in $\boldsymbol{\mathcal{W}}$ in order to explain/model the data as well as possible. The quality of the fitting to the data is quantified by (4.23).

4.2.2 Model Training

The GTM-HMTM can be trained using the EM algorithm as previously considered in the setting of mixture models in sections 3.2.1 and 3.3.3, and in the setting of a single HMTM in section 3.4.2. Regarding the GTM-HMTM extension, the E-step stays essentially the same as in section 3.4.2, while the M-step changes because of the different parametrisation used (parameters in this setting are produced by the constrained RBF mapping).

Since the EM terrain is by now familiar, we move swiftly with the introduction of hidden variables $z \in \mathcal{Z}$:

$$z_c^{(n)} = \begin{cases} 1, & \text{if tree } \boldsymbol{y}^{(n)} \text{ was generated by model } c; \\ 0, & \text{otherwise.} \end{cases}$$

$$z_{c,k}^{(n,u)} = \begin{cases} 1, & \text{given tree } \boldsymbol{y}^{(n)} \text{ was generated by model } c, \\ & \text{node } u \text{ was in state } k; \\ 0, & \text{otherwise.} \end{cases}$$

$$z_{c,k\to l}^{(n,u)} = \begin{cases} 1, & \text{given tree } \boldsymbol{y}^{(n)} \text{ was generated by model } c, \\ & \text{node } u \text{ was in state } l \text{ and its parent} \\ & \text{in state } k; \\ 0, & \text{otherwise.} \end{cases}$$

Taking into account $\mathcal{Z}$ and using (4.23), we write the (scaled) complete-data log-likelihood:

$$\begin{aligned}\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D}) \quad \propto \quad & \sum_{n=1}^{N}\sum_{c=1}^{C} z_c^{(n)} \Bigg[z_{c,k}^{(n,1)} \sum_{k=1}^{K} \log p(q_1 = k|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \\ + \quad & \sum_{u=2}^{U_n}\sum_{k=1}^{K}\sum_{l=1}^{K} z_{c,k\rightarrow l}^{(n,u)} \log p(q_u = l|q_{\rho(u)} = k, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \\ + \quad & \sum_{u=1}^{U_n}\sum_{k=1}^{K} z_{c,k}^{(n,u)} \log p(\boldsymbol{o}_u^{(n)}|q_u = k, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \Bigg]. \end{aligned} \tag{4.25}$$

The expectations of the hidden variables at iteration i, given dataset $\mathcal{D}$ and current weights $\boldsymbol{\mathcal{W}}^{(i)}$, are calculated as follows:

$$E[z_c^{(n)}|\mathcal{D}, \boldsymbol{\mathcal{W}}^{(i)}] = p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}, \boldsymbol{\mathcal{W}}^{(i)}), \tag{4.26}$$

$$E[z_{c,k}^{(n,u)}|\mathcal{D}, \boldsymbol{\mathcal{W}}^{(i)}] = p(q_u = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}^{(i)}), \tag{4.27}$$

$$E[z_{c,k\rightarrow l}^{(n,u)}|\mathcal{D}, \boldsymbol{\mathcal{W}}^{(i)}] = p(q_u = l, q_{\rho(u)} = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}^{(i)}). \tag{4.28}$$

These expectations allow us to calculate at iteration i the (scaled) expected complete-data log-likelihood:

$$\begin{aligned} E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\mathcal{W}}^{(i)}] \quad \propto \quad & \sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}, \boldsymbol{\mathcal{W}}^{(i)}) \\ \times \quad & \Bigg[\sum_{k=1}^{K} p(q_1 = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}^{(i)}) \log p(q_1 = k|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \\ + \quad & \sum_{u=2}^{U_n}\sum_{k=1}^{K}\sum_{l=1}^{K} p(q_u = l, q_{\rho(u)} = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}^{(i)}) \\ \times \quad & \log p(q_u = l|q_{\rho(u)} = k, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \\ + \quad & \sum_{u=1}^{U_n}\sum_{k=1}^{K} p(q_u = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}^{(i)}) \log p(\boldsymbol{o}_u^{(n)}|q_u = k, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \Bigg]. \end{aligned} \tag{4.29}$$

In the M-step, the derivatives of the expected log-likelihood are calculated with respect to the parameters of the model:

$$\frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\mathcal{W}}^{(i)}]}{\partial \boldsymbol{W}^{(\boldsymbol{\pi})}}, \frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\mathcal{W}}^{(i)}]}{\partial \boldsymbol{W}^{(\boldsymbol{B}_k)}}, \frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \boldsymbol{\mathcal{W}}^{(i)}]}{\partial \boldsymbol{W}^{(\boldsymbol{\psi}_k)}}.$$

Optimum values for the parameters in $\boldsymbol{\mathcal{W}}$ are calculated by setting the above derivatives to

zero. This results in the following update equations:

Element $w_{j,m}$ in matrix $\boldsymbol{W}^{(\boldsymbol{\pi})}$:

$$\begin{aligned}\frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}]}{\partial w_{j,m}} &= \sum_{n=1}^{N}\sum_{c=1}^{C}\phi_m(\boldsymbol{x}_c)p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \\ &\times \Big(p(q_1=j|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) - p(q_1=j|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})\Big), \quad (4.30)\end{aligned}$$

Element $w_{j,m}^{(r)}$ in matrix $\boldsymbol{W}^{(\boldsymbol{B}_r)}$:

$$\begin{aligned}\frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}]}{\partial w_{j,m}^{(r)}} &= \sum_{n=1}^{N}\sum_{c=1}^{C}\phi_m(\boldsymbol{x}_c)p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \\ &\times \sum_{u=2}^{U_n}\Big(p(q_u=j,q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \\ &- p(q_u=j|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})p(q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})\Big), \quad (4.31)\end{aligned}$$

Element $w_{j,m}^{(r)}$ in matrix $\boldsymbol{W}^{(\boldsymbol{\psi}_r)}$:

$$\begin{aligned}&\frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}]}{\partial w_{j,m}^{(r)}} = \\ &\sum_{n=1}^{N}\sum_{c=1}^{C}\phi_m(\boldsymbol{x}_c)p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)})\sum_{u=1}^{U_n}p(q_u=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})\boldsymbol{e}_k\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}_u^{(n)}-\boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c)). \quad (4.32)\end{aligned}$$

where $\boldsymbol{e}_k$ is defined as the row unit-vector which has all elements equal to zero apart from entry k equal to 1, $\boldsymbol{\Sigma}_k$, $k = 1, 2, ..., K$ are the covariance matrices of the state-conditional Gaussian emissions. Detailed derivations are found in Appendix B.

After training the model, we can represent each data item $\boldsymbol{y}^{(n)}$ with a point $proj(\boldsymbol{y}^{(n)})$ in the latent space given by the expectation of the posterior distribution over all latent points $\boldsymbol{x}_c$:

$$proj(\boldsymbol{y}^{(n)}) = \sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}})\boldsymbol{x}_c. \quad (4.33)$$

Regarding the covariance of the emission distribution, we noticed that higher quality models were obtained when instead of direct modelling of the covariance through the map Γ, the covariance was calculated in the fashion of [Bishop et al., 1998], at the end of each M-step using standard update equations:

$$\Sigma_{k,hj}^{(c)} = \frac{\sum_{n=1}^{N} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}, \boldsymbol{\mathcal{W}}^{(i)}) \sum_{u=1}^{U_n} p(q_u = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}^{(i)})(o_{u,h}^{(n)} - \mu_{k,h}^{(c)})(o_{u,j}^{(n)} - \mu_{k,j}^{(c)})}{\sum_{n=1}^{N} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}, \boldsymbol{\mathcal{W}}^{(i)}) \sum_{u=1}^{U_n} p(q_u = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}^{(i)})}, \quad (4.34)$$

where $i, j = 1, 2, ..., d$ index the elements of the mean and label vectors $\boldsymbol{\mu}$ and $\boldsymbol{o}$ respectively, as well as the elements of the covariance matrix $\boldsymbol{\Sigma}$. The problem with directly modelling covariance seems to be the same one also encountered in mixtures of Gaussians. Once (co)variances of a component become small, overfitting may occur as the component may concentrate on modelling only a few (perhaps even one) data points. It is very difficult to alleviate this pathological situation, especially in the absence (which is the case here) of an initialisation procedure, which forces us to initialise parameters, including covariance, randomly.

After the training, to smooth the covariance structure of local HMTMs addressed by the latent points, we recalculated the covariance matrices using the following scheme: Covariance matrix $\boldsymbol{\Sigma}_k(\boldsymbol{x})$ of the HMTM addressed by $\boldsymbol{x}$ is expressed as a convex combination of the corresponding covariance matrices[2] $\boldsymbol{\Sigma}_k^{(c)}$ of HMTMs addressed by latent centres $\boldsymbol{x}_c$, $c = 1, 2, \ldots, C$:

$$\boldsymbol{\Sigma}_k(\boldsymbol{x}) = \sum_{c=1}^{C} \nu_c(\boldsymbol{x}) \boldsymbol{\Sigma}_k^{(c)}, \quad (4.35)$$

where

$$\nu_c = \frac{\exp(-\beta \|\boldsymbol{x} - \boldsymbol{x}_c\|)}{\sum_{c'=1}^{C} \exp(-\beta \|\boldsymbol{x} - \boldsymbol{x}_{c'}\|)}, \quad (4.36)$$

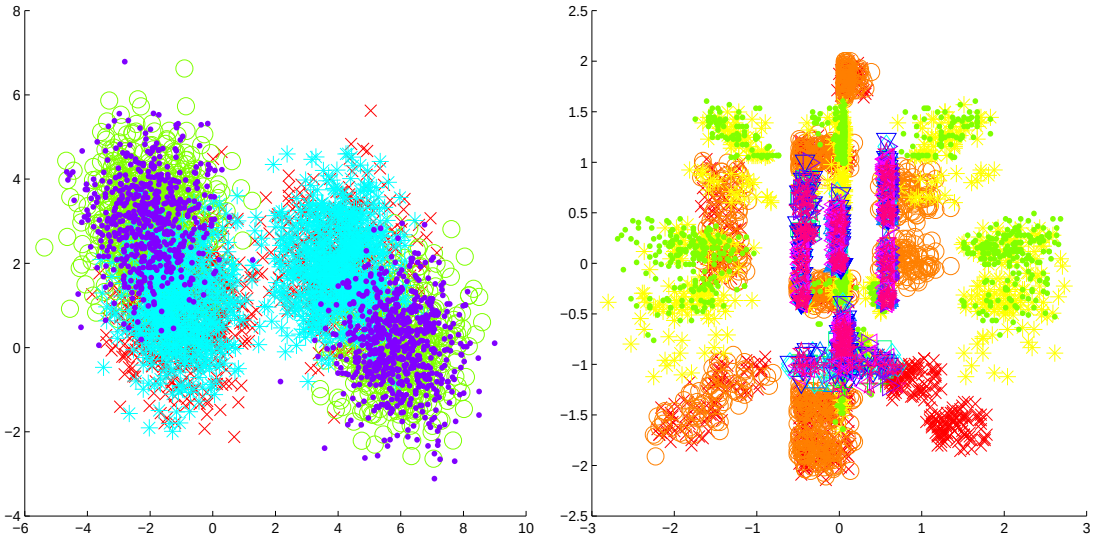
and $\|\cdot\|$ denotes the Euclidean norm on $\mathcal{V}$. The parameter $\beta > 0$ quantifies to what degree local neighbourhoods of $\boldsymbol{x}$ are considered.

Here we have set $\beta = 10$, but we have found that the visualisation plots were similar for a wide range of β values. In practice, compared to the obvious choice of directly parameterising the covariance matrices through a smooth mapping from the latent space, we found that this scheme leads to superior models (viewed as density estimators and evaluated on a hold out set) and hence better visualisation plots.

[2]Note that a convex combination of symmetric positive definite matrices is again a symmetric positive definite matrix.

Table 4.1: Parameters of HMTMs for creating the toy dataset. Variance was fixed to $\sigma^2 = 1$.

Class	Initial prob	Transition prob	Means of emissions
HMTM 1	0.7 0.3	0.9 0.1 0.1 0.9	$\begin{pmatrix} -1.0 \\ 1.0 \end{pmatrix}$ $\begin{pmatrix} 4.0 \\ 2.0 \end{pmatrix}$
HMTM 2	0.7 0.3	0.9 0.1 0.1 0.9	$\begin{pmatrix} -2.0 \\ 3.0 \end{pmatrix}$ $\begin{pmatrix} 6.0 \\ 0.0 \end{pmatrix}$
HMTM 3	0.7 0.3	0.1 0.9 0.9 0.1	$\begin{pmatrix} -1.0 \\ 1.0 \end{pmatrix}$ $\begin{pmatrix} 4.0 \\ 2.0 \end{pmatrix}$
HMTM 4	0.7 0.3	0.1 0.9 0.9 0.1	$\begin{pmatrix} -2.0 \\ 3.0 \end{pmatrix}$ $\begin{pmatrix} 6.0 \\ 0.0 \end{pmatrix}$



(a) (b)

Fig. 4.3: Labels of toy (a) and TPB (b) dataset. Each marker style indicates class membership of the tree to which each label belongs.

4.3 Experimental Results for GTM-HMTM

4.3.1 Datasets

We have used three datasets in our experiments. The first dataset is an artificial toy dataset produced by sampling from 4 HMTMs with 2 hidden states with two-dimensional Gaussian emissions of fixed spherical variance, each corresponding to one artificial class. The dataset was populated 320 data items by generating 80 samples for each class. All patterns have the topology of a binary tree with 15 nodes. The parameters of the models were set in such a way as to ensure that it would be impossible to distinguish the classes from the observations alone, without taking

Table 4.2: Classes in TPB dataset.

Class	Symbol	Description
A	◯	Policemen with the lowered left arm
B	x	Policemen with the raised left arm
C	*	Ships with two masts
D	•	Ships with three masts
E	△	Houses with one upper right window
F	▽	Houses with upper left and lower left window
G	◁	Houses with two upper windows
H	▷	Houses with lower left and upper right window
I	⋆	Houses with three windows
J	□	Houses with one lower left window
K	+	Houses with no windows
L	♢	Houses with one upper left window

into account the underlying tree structure. A plot of two-dimensional observations of all the nodes for all trees is presented in Fig. 4.3(a). The parameters of the HMTMs are summarised in table 4.1.

The second dataset consists of benchmark images produced by the *Traffic Policeman Benchmark* (TPB) software [Hagenbuchner and Tsoi, 1999]. The same software was used to produce a dataset to demonstrate the functionality of SOM for Structured Data (SOMSD) in [Hagenbuchner et al., 2003]. This software provides an artificial domain for evaluating learning algorithms that process structured patterns. It produces images that resemble traffic policemen, houses and ships of different shape, size and colour that are products of a rule based grammar. Three sample images of each type are illustrated in Fig. 4.5. Connected components in each image have a parent-child relationship, the object located lower and closer to the left edge being the parent (i.e. the images must be interpreted bottom-up, left to right). In Fig. 4.5(d), 4.5(e) and 4.5(f) tree representations of the sample images corresponding to Fig. 4.5(a), 4.5(b) and 4.5(c) are displayed. TPB produces general acyclic graph structures, but we restricted it to generate only images expressed as trees. Each node in each tree is labelled with a two-dimensional vector. This two-dimensional vector is a pair of coordinates for the centre of gravity of the component that node stands for. The dataset defines 12 classes that are presented in table 4.2. For each class 50 samples were generated resulting to a dataset of 600 samples. Also, a validation set of 84 tree data items was produced by generating 7 samples for each class. Fig. 4.3(b) is a plot of the labels of trees in the dataset. This illustrates what the observed data look like if the tree

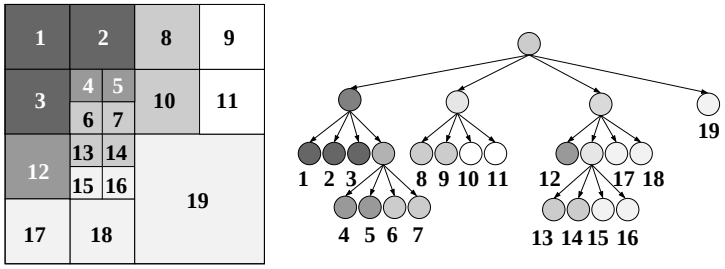


Fig. 4.4: Example of an image represented as a quadtree. Numbers show the association between quadrants in the image and nodes in the quadtree.

structure is ignored.

The third dataset consists of images interpreted as *quadtrees*. A quadtree is a data structure used amongst other things for storage of images [Samet, 1990]. It is a 4-regular tree $\boldsymbol{y}$, i.e. each parent node u has exactly 4 children $v_r \in ch(u), r = 1, \ldots, 4$ (apart from the leaf nodes). An example of a quadtree is displayed in Fig. 4.4. A quadtree $\boldsymbol{y}$ stores an image in a recursive manner; the root note, node u_1, represents the entire image. At the first level of the recursion, the image is partitioned into four equal square quadrants. At the first level of quadtree $\boldsymbol{y}$, each node $v \in ch(u_1)$ represents a quadrant, and is labelled by a scalar that expresses the mean colour intensity of the quadrant. At the next level of the recursion, each quadrant is partitioned further into four quadrants and their mean colour intensities are stored as labels in the nodes at the second level of quadtree $\boldsymbol{y}$. Partitioning continues in this fashion either until a quadrant becomes a single pixel, or when a certain criterion is met. Such a criterion can be a function of the relative change in mean colour intensity between a node u and its parent $\rho(u)$. We note that quadtrees can represent only images of a dimension that is a power of 2 since images are progressively divided into smaller square regions. Other images must be padded with extra pixels or resized in order to become of appropriate dimension.

The images used here are taken from the Amsterdam Library of Object Images (ALOI) database [Geusebroek et al., 2005]. We selected 72 images of a single object, a rubber duck, photographed from different viewing angles. The dataset was divided into a training and validation set of 48 and 24 images respectively. The images were created in [Geusebroek et al., 2005] by successively rotating the object by an angle of 5° degrees and photographing it from each angle of rotation. The images are colour images of dimension 192×144 (pixels). We converted the images into grayscale and resized them into square images of dimensions 64×64. The number of grayscale levels was then further reduced to 4 levels, which still allows enough detail to be discerned relative to the original images. The values of the 4 quantisation levels were determined by first collecting the grayscale intensities of all pixels from all images and then

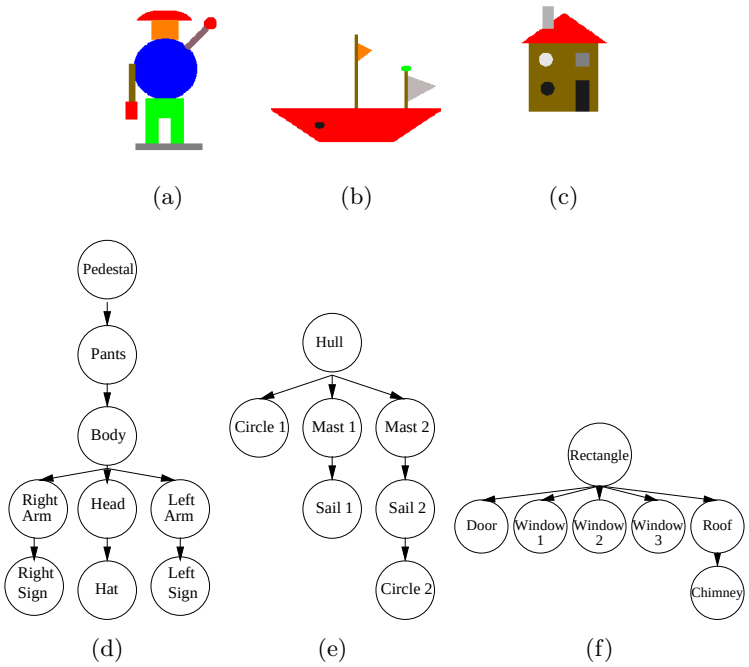


Fig. 4.5: Sample images from TPB in (a), (b), (c) and their corresponding tree representations in (d), (e), (f).

using the k-means algorithm to select 4 centres in the space of pixel intensities.

All three datasets were normalised in each dimension to zero mean and unit standard deviation.

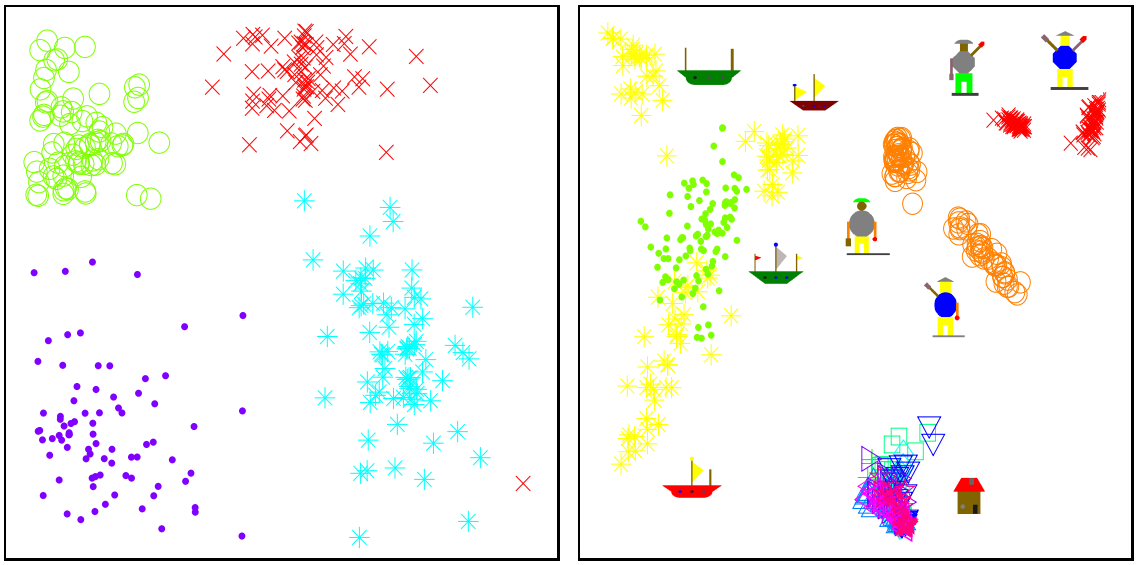
4.3.2 Training

The lattice was a 10x10 regular grid (i.e. $C = 100$) and the RBF network consisted of $M = 17$ basis functions; 16 of them were Gaussian radial basis functions of variance $\sigma^2 = 1$ centred on a 4x4 regular grid and one was a constant function $\phi_{17}(\boldsymbol{x}_c) = 1$ intended to serve as a bias term (analogous to the bias in neural networks).

The state-conditional emission probability distributions were modelled as two-dimensional spherical Gaussians. During training the emission covariance was updated according to (4.34). Parameters were initialised randomly with uniform distribution in $[-1, 1]$.

We employed scaled conjugate gradient for optimising the cost function (4.29). The gradient was calculated using (4.30), (4.31) and (4.32).

In [Durand et al., 2004] it is mentioned that the complexity of the upward-downward recursion for processing a single tree $\boldsymbol{y}$ is $\mathcal{O}(U_{\boldsymbol{y}} K^2)$. In the GTM-HMTM the recursion must be repeated N times for each tree $\boldsymbol{y} \in \mathcal{D}$ and C times for each latent point $\boldsymbol{x}_c \in \mathcal{V}, c = 1 \dots C$.



(a) (b)

Fig. 4.6: Visualisation of toy (a) and TPB (b) dataset using GTM-HMTM.

Hence, the complexity of the E-step is estimated as $\mathcal{O}(NC\bar{U}K^2)$ where $\bar{U}$ is the average number of nodes of a tree in $\mathcal{D}$. The complexity of the M-step is that of the scaled conjugate optimisation which is calculated as $\mathcal{O}(2W^2)$ in [Moller, 1993], W being the number of free parameters to be optimised. The number of free parameters in GTM-HMTM is $W = MK(K + d + 1)$ which is the number of elements on matrix $\boldsymbol{\mathcal{W}}$.

In practice, training times for learning a topographic map for each dataset were in hours 2 for the toy dataset, 18 for TPB and 15 for the quadtrees when run on a machine equipped with an Athlon XP 3000+ CPU and 512MB of memory. Algorithms were implemented in MATLAB (version 7.3) and were partially vectorised.

4.3.3 Results and Discussion

In Fig. 4.6(a) we see topographic organisation achieved by the GTM-HMTM of the toy dataset for $K = 2$. The covariance of the emission distribution was initially set to $\Sigma_k = 2I$ for both states $k = 1, 2$ where I stands for the identity matrix. We also tried initialising it with $\Sigma_k = 2I, 3I, 5I$ with similar success. Each point on the plot represents an input pattern (tree) and four different markers correspond to the four generative classes used to construct the data set. Training is completely unsupervised and class markers are used only after the training when plotting the projections. A clear topographic organisation of classes has been achieved - there is an evident trend of patterns of the same class to belong to the same cluster.

Fig. 4.6(b) shows the visualisation of the traffic policeman benchmark (TPB) data set pro-

duced by GTM-HMTM with $K = 2$. The initial covariance matrix for the emission distribution was set to $\Sigma_k = 2I$ for both states $k = 1, 2$. We also tried initialising the covariance matrix with $\Sigma_k = 1I, 3I$ which yielded similar results and $\Sigma_k = 0.5I$ which failed to achieve the same level of topographic organisation. Moreover, we attempted training for $K = 3, 4$, but with suboptimal results. One problem that makes training difficult is that as the number of states (and consequently the number of free parameters of the model) increases, it becomes more vital for an EM trained model to use a good initialisation strategy for the weights. In GTM the initial weights are determined by the linear projection space obtained through principal component analysis [Bishop et al., 1998]. In our case we do not have such a luxury. One way of dealing (at least to certain degree) with the initialisation problem would be to abandon the EM framework and adopt a more stable parameter fitting strategy (e.g. Bayesian).

In Fig. 4.6(b), next to each cluster a representative image is displayed. The model has clearly achieved a level of topographic organisation. It is interesting to note the emerging sub-clusters. Class $\times$ has been split into two sub-clusters, one with policemen with the right arm lowered and one with the right arm raised. The same has happened for class $\bigcirc$ which has been divided into policemen with the right arm lowered and policemen with the arm raised. The sub-clusters of ships are also interesting as not only has class $*$ been divided into three sub-clusters, but the sub-clusters that surround class $\bullet$ possibly indicate how the classes are related. Thus, class $\bullet$ seems to act as a "link" between the three discovered sub-clusters; class $\bullet$ represents ships with three masts, while the three sub-clusters around class $*$ are composed of ships with either the two masts, with either the left, centre or right mast missing. Nevertheless, the model has not been successful in the visualisation of the classes representing houses. No clusters have been formed as all classes have been merged into one big cluster representing a super-class of all the images of houses. One possible explanation for this inability of discriminating between the classes of houses, is the shallow tree representation of houses; typically they are shorter than ships and traffic-policemen structures.

In Fig. 4.7 the underlying state transitions are visualised. The plot is organised as a grid of $K \times K = 2 \times 2$ state transition matrices $p(q_u = l | q_{\rho(u)} = k)$, each transition matrix corresponding to an underlying local noise model (HMTM). Topographic organisation of local noise models with respect to their transition structure is evident in Fig. 4.7 as state transitions vary smoothly with their "latent space addresses". In Fig. 4.7 we see that state 1 acts as a "trap" state for the entire plot, that is if the model visits state 1, it is extremely unlikely for it to ever visit state 2. Regarding transitions from state 2 we observe a more interesting behaviour. A strong tendency for self-loops in state 2 is observed at the upper-left and bottom-left corner of Fig. 4.7. However,

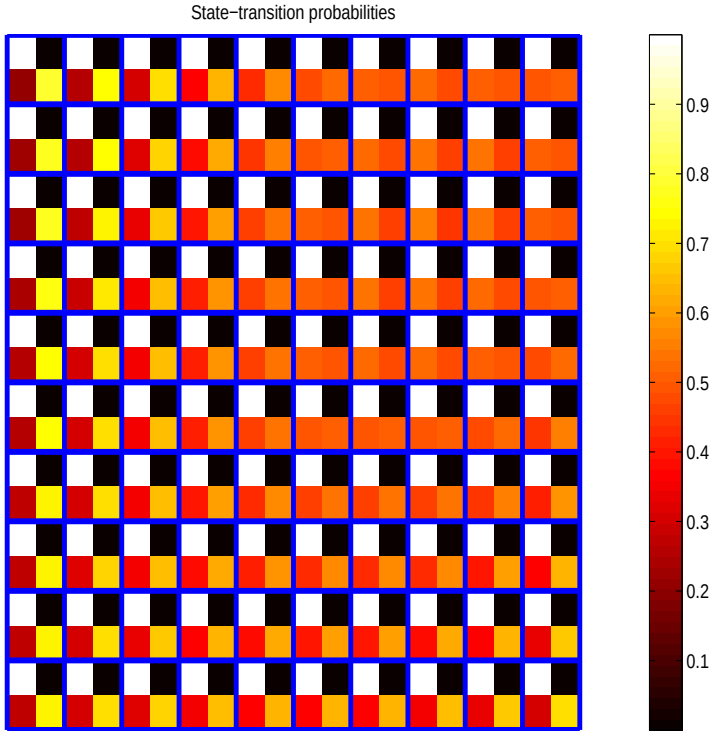


Fig. 4.7: TPB task: grid of 2×2-state transition matrices corresponding to the local HMTMs underlying the visualisation plot.

this behaviour gradually changes as we move towards the centre of the map; transitions to state 2 progressively lose their strength benefiting transitions to state 1. Around the centre of the map transitions to state 1 narrowly dominate transitions to state 2. Moving further towards the upper right part of the plot, transitions to state 1 and 2 become almost equally likely. Moving from the centre towards the bottom-right corner, transitions to state 2 regain their power, albeit not to the same strength as in the upper-left and bottom-left corner of the plot.

The respective plot for the initial probabilities is not presented, as a particularly simple structure has emerged as a result of the GTM-HMTM training; the initial probability vector of all models is practically equal to $\boldsymbol{\pi} \approx [0,\ 1]^T$. Thus, effectively all models pick the second state as their starting state, $q_1 = 2$.

In Fig. 4.8 the underlying means of the emissions are visualised. This plot is organised as a grid of subplots. Each subplot presents the space of emissions $\mathbb{R}^d$, where labels $\boldsymbol{o}_u$ reside, in which the means for states $k = 1$ and $k = 2$ marked with circles and crosses respectively. Evidently, the means of the emissions are topographically organised as well as the state transitions, as the positions of means change gradually as we move in the plot. We note that images in the TPB dataset are interpreted bottom-up (see Fig. 4.5), and that x-coordinates of the labels decrease

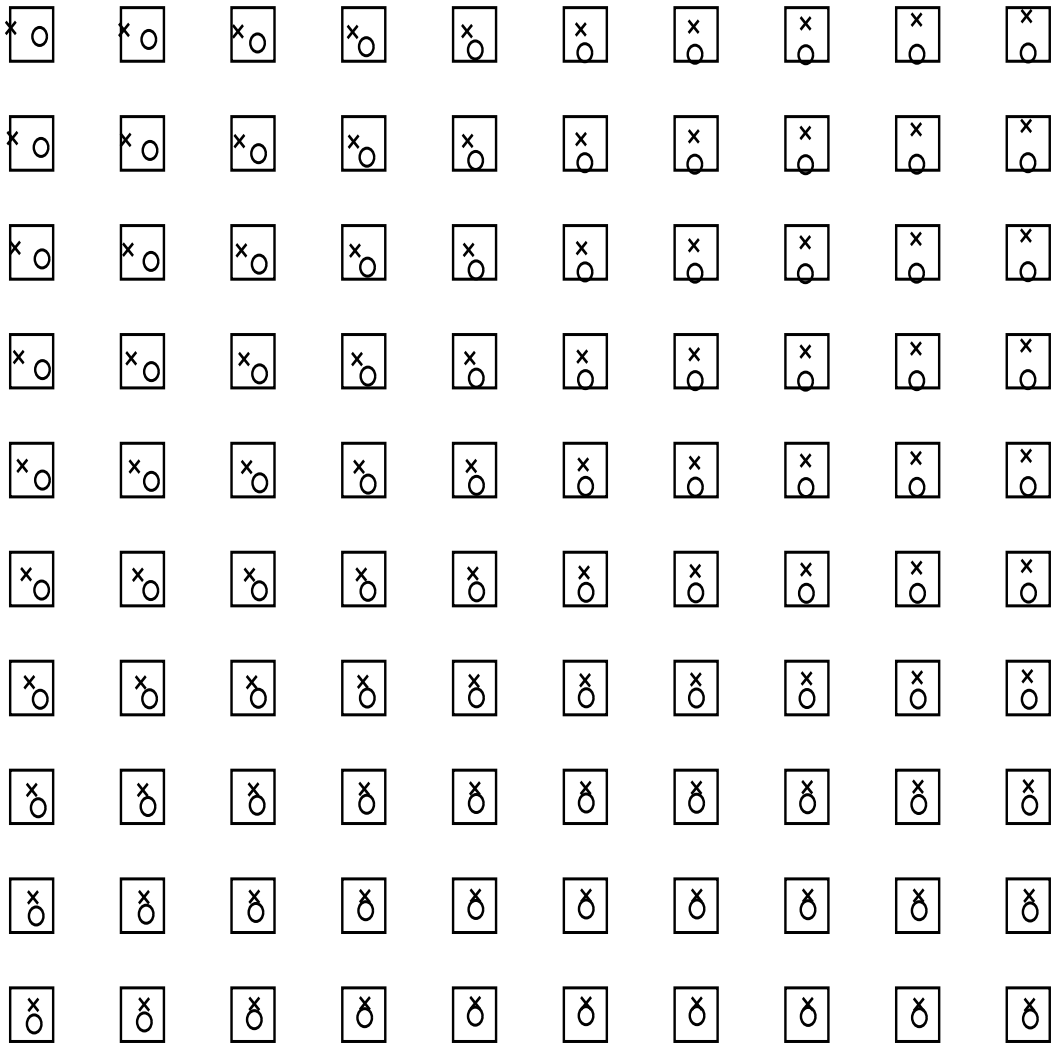


Fig. 4.8: TPB task: means of emissions for states $k = 1, 2$ corresponding to the local HMTMs. Means corresponding to states 1 and 2 are marked with circles and crosses respectively.

leftwards while y-coordinates decrease upwards. Thus, components located at the lower part of the TPB images have higher y-coordinates than components located closer to the top of the TPB images. We observe that since state 2 is effectively the starting state for all models (since $\boldsymbol{\pi} \approx [0,\ 1]^T$) and since images are interpreted bottom-up, the mean for state 2 naturally has a greater y-coordinate than the mean for state 1 in the entire plot. We also observe the following three general behaviours in the plot. The means close to the upper-left corner of the plot lie far apart in the x-axis, while being close in the y-axis. This behaviour progressively changes as we move towards the upper-right corner of the plot, where the means have similar x-coordinates but are distant in the y-axis. Moving towards the bottom-centre and bottom-right regions of the plot we notice that the means approach each other. This behaviour reflects the nature of

the data points (trees) mapped in these particular regions of the latent space. In particular, the ship-classes reserve the left part of the visualisation plot in Fig. 4.6(b). As it can be seen in Fig. 4.5, ships are generally "wide" and "short" structures. Policemen are concentrated at the right upper part of Fig. 4.6(b), and are generally "narrow" and "thin"(see Fig. 4.5), while houses are clustered densely at the bottom-centre of Fig. 4.6(b) and appear to be relatively "compact" (see Fig. 4.5). In order to confirm these observations, we measured the variance for the three classes of ships, policeman and houses. We found that the variance was $1.83, 0.69, 0.14$ in the x-axis and $0.58, 1.53, 0.42$ in the y-axis for the three classes respectively.

Inspecting Fig. 4.7 in conjunction with Fig. 4.8 we make the following observations. In general, the mean for state 1 concentrates more on modelling the labels of lower y-coordinates, while the mean for state 2 seems to concentrate more on the labels of higher y-coordinates. The classes of ships reserve the area that corresponds to the left area in Fig. 4.7 of self-loops for state 2, thus favouring the projection of "short" classes[3]. Furthermore, the upper right area of the latent space, in Fig. 4.6(b), is reserved for the policemen classes, which are "tall" structures. As noted, this corresponding area in the state-transition plot of Fig. 4.7 is where transitions from state 2 to states 1 and 2 become almost equally likely, thus favouring such "tall" structures. Of course, if transitions from state 2 to state 1 were further strengthened at the expense of transitions from state 2 to itself, the projection of the policemen classes to the corresponding area would be favoured even further. This particular area in Fig. 4.7 is the most favourable for the projection of the policeman classes with respect to other regions of the latent space. Finally, the respective area of the house classes in Fig. 4.7 corresponds to the area where a strong tendency for self-loops for state 2 occurs, that favours the mapping of "short" structures. Clearly, despite of the similarity in the state-transition probabilities in the respective areas of the ship and house classes, the two classes are projected in well separated areas due to the different underlying structure of the means. The model-based nature of the visualisation plots brings a transparency of GTM-HMTM in analysing and understanding of how the data items are organised in the visualisation plot in Fig. 4.6(b).

We also trained GTM-HMTM on the dataset of quadtrees. We set $K = 3$ and the variance of the one-dimensional emissions equal to 1.0. However during training, GTM-HMTM displayed numerical problems that prevented us from using the dataset at the 64×64 resolution that we specified earlier. Thus, we reduced the images from 64×64 to 16×16 pixels. The results for GTM-HMTM on the quadtree dataset are displayed in Fig. 4.9. Unfortunately, although a certain level of topographic organisation is evident, the model does not seem to be particularly

[3]recall that the values of y-coordinate in TPB data increase in a top-down direction.

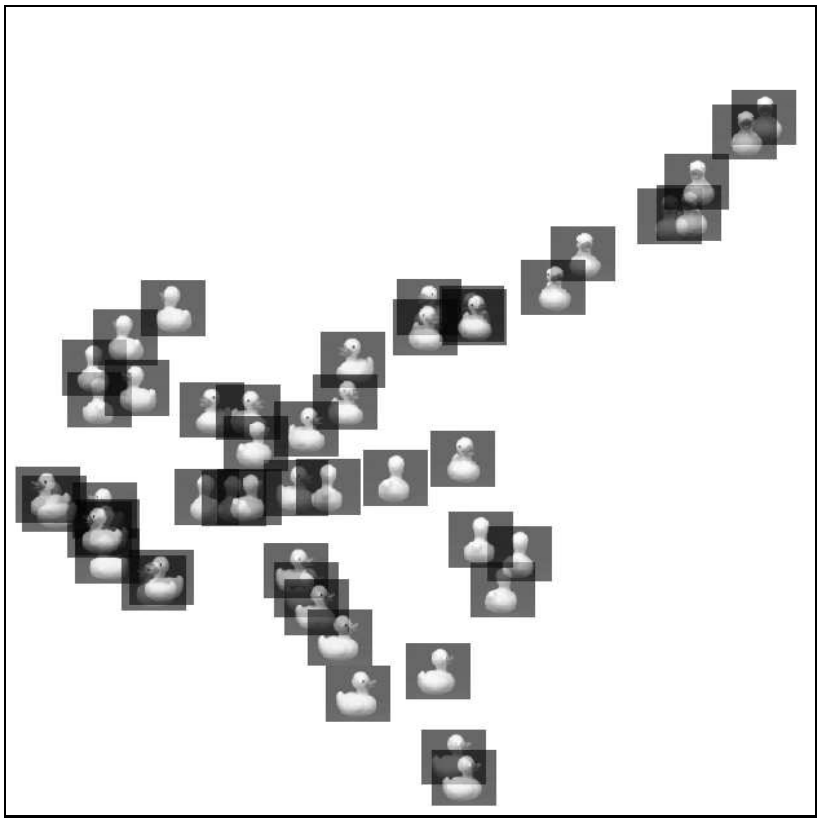


Fig. 4.9: Visualisation of reduced resolution quadtree dataset (16×16) for GTM-HMTM. Images are plotted as transparent to allow visibility of overlapping ones.

successful at this task. We also trained GTM-HMTM setting $K = 2, 4$ in combination with different values for the initial variance such as $0.1, 0.5, 0.8$ with less success. We note certain trends such as the presence of images at the bottom of the plot of ducks facing to the right, while at the centre-left we come across images facing to the left. The top right is dominated to images of frontal views. Finally, close to the centre and slightly to the left, we note an overlapping of images of different orientations that have not been successfully organised on the map.

Further insight regarding the topographic organisation for the quadtree dataset can be gained by observing the plots for the state transitions in Fig. 4.10 and the means of the emissions in Fig. 4.11. The state transitions are very similar across the entire plot and only subtle variations are noticeable. All three plots for the means exhibit a very similar structure, with abrupt changes close to the centre of the respective plots. These observations suggest that the underlying local models are very similar in terms of transition probabilities, and that it is the means that mostly drive the topographical organisation. The abrupt changes noted in the plots of the means, seem to be related to the overlapping of images of different orientations, noted at about the same location in Fig. 4.9 (close to the centre of the plot).

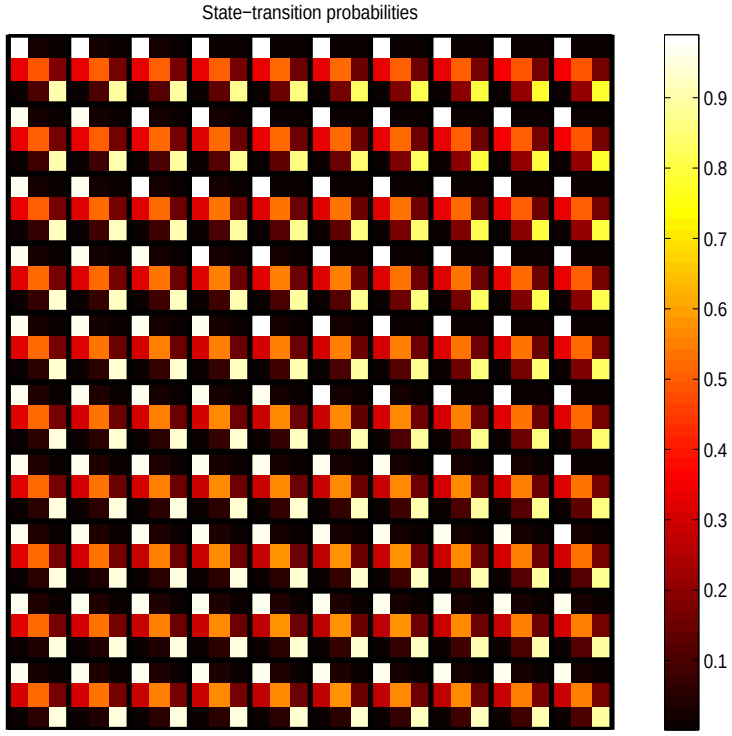
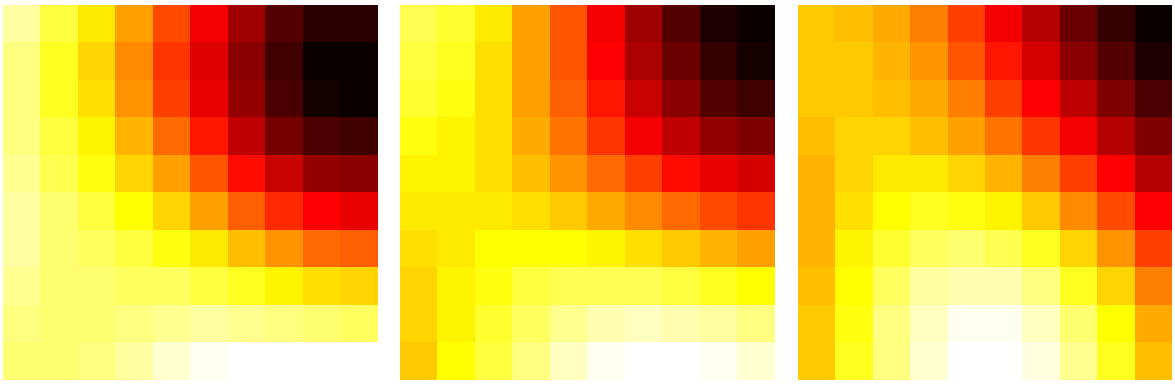


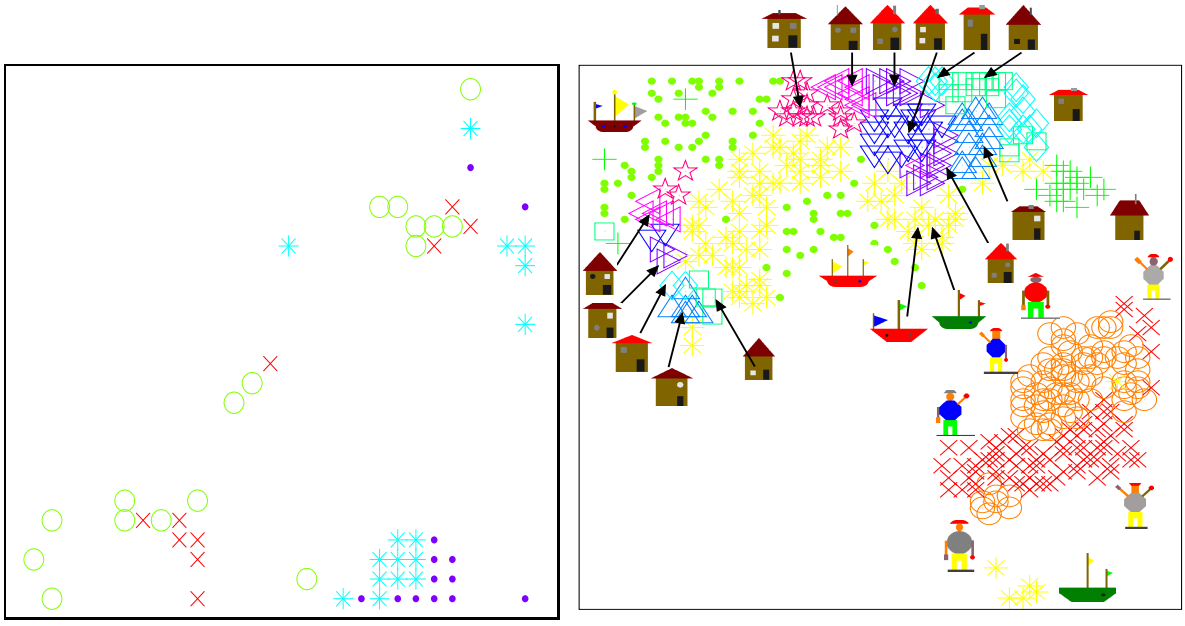
Fig. 4.10: Quadtree task: grid of 3×3-state transition matrices corresponding to the local HMTMs underlying the visualisation plot.



(a) (b) (c)

Fig. 4.11: Quadtree task: means of emissions of GTM-HMTM for quadtree dataset. Each plot corresponds to a state. The plots are coloured as heat maps with the rank of colours ranging from white to yellow to red to black corresponding from higher to lower values.

As a comparison we also present the results obtained by using SOMSD on the three datasets. We tried numerous parameter settings for SOMSD all with rectangular lattices, Gaussian neighbourhood functions, 600 training iterations and picked the best results for the toy and TPB datasets where class information is provided, according to the criterion described below. For the toy dataset we found that the best parameters were a lattice of dimensions 28×28, a learning



(a) (b)

Fig. 4.12: Visualisation of toy (a) and TPB (b) dataset using SOMSD.

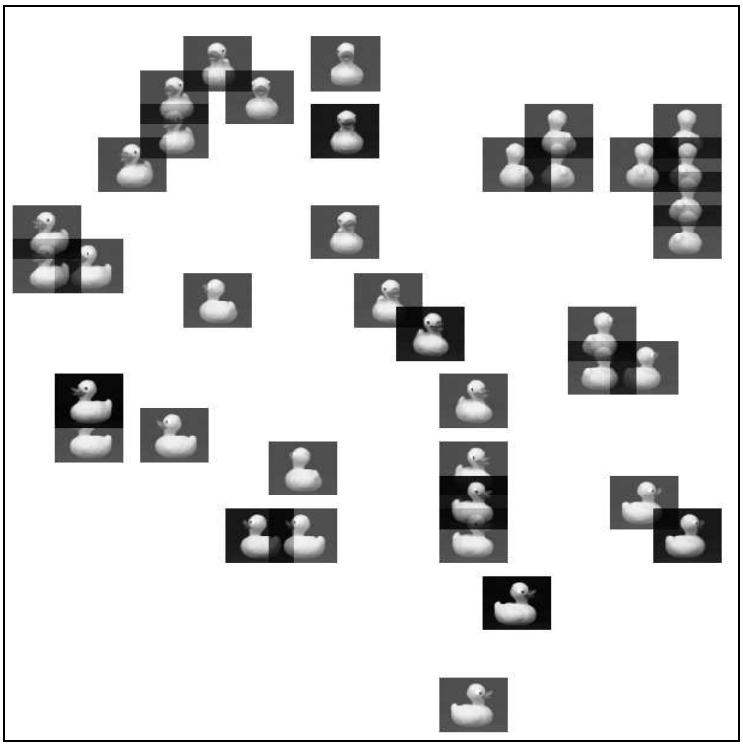


Fig. 4.13: Visualisation of reduced resolution quadtree dataset (8×8) for SOMSD.

rate of 0.5, an initial radius of 5 and weighting coefficients of $\mu_1 = 0.99$ and $\mu_2 = 0.01$. For the TPB dataset we chose a network of dimensions 114×87, a learning rate of 1.5, an initial radius of 60 and weighting coefficients of $\mu_1 = 0.01$ and $\mu_2 = 0.99$. Finally for the quadtree dataset,

the parameters were a network of dimensions 90×90, a learning rate of 0.5, initial radius of 90 and weighting coefficients of $\mu_1 = 0.05$ and $\mu_2 = 0.95$. By inspecting the plots we can see that GTM-HMTM is better at the toy dataset, while SOMSD is better at the TPB dataset as it manages to distinguish between all of the classes, especially the classes of houses that are problematic in GTM-HMTM. This is interesting, because SOMSD seems to be more sensitive than GTM-HMTM to data items of shallow structure. On the other hand, SOMSD does not discover the sub-classes that GTM-HMTM does for the policemen and ships. Regarding the quadtree dataset, although we tried numerous parameter settings we could not obtain a good result for the same dataset of 16×16 pixel images. Nevertheless, when we further reduced the dimensions of the images down to 8×8 pixels, SOMSD was able to a achieve good topographic organisation, displayed in in Fig. 4.13, indicating that the transformed images preserve sufficient information. However, SOMSD does not seem to utilise the entire map when projecting the quadtree data, as it does for the toy and TPB dataset (the same problem also appeared when training with smaller maps). Thus, in Fig. 4.13 only the region of the map containing projections is displayed.

The toy data set may be biased towards GTM-HMTM, but still, SOMSD was not able to cluster the trees in a fashion reflecting the organisation of the underlying generative process. This raises an important point we would like to stress. Of course, there is no single best model for topographic organisation of data of a given form. This issue is even more pronounced in the case of unsupervised learning in structured domains, where for models such as SOMSD a clear cost functional being minimised during the parameter fitting process is missing. Besides not knowing exactly what the model is optimised for, there is an additional difficulty: recursive models such as SOMSD are non-autonomous dynamical systems that can be difficult to understand. But without a clear understanding of the underlying dynamics, we can never know exactly what is driving topographical organisation of the projections. As a consequence, given a new tree, it might be possible to guess where its image on the SOMSD map will lie, but understanding the process of its formation will remain problematic. Consequently, it is difficult to grasp the structure of a trained topographic map on a deeper level - one is forced to produce only verbal descriptions.

In contrast, a clear model-based formulation of GTM-HMTM enables us not only to analyse and understand the trained model (and hence understand the organisation of the map in terms of organisation of local prototype HMTM noise models), but also to understand exactly what kinds of data our model is suitable for. It is also important to understand that the class of noise models (in our case HMTM) inherently dictates along what lines will the data projections/representations be organised on the visualisation plot. Close regions on the computer

screen (latent space $\mathcal{V}$) will correspond to "close" noise models (HMTMs) and hence trees will be organised on the map with respect to how closely they adhere to different HMTMs defined by different regions on the map.

Because of the absence of a clear cost function, the performance of SOMSD was measured in [Hagenbuchner et al., 2003] as the accuracy of classification of data into known classes (the class information was not used during the training) using data representations on the map. After the map formation, a secondary hold-out test dataset was used. Items from the test set were represented on the trained map and each test item was predicted to have the class label of its closest neighbour (from the training set) on the map. The accuracy was then defined as the percentage of correctly classified test points. The results of this measure on the toy dataset were 90% and 60% for GMT-HMTM and SOMSD respectively. The results were reversed as for the TPB dataset GMT-HMTM and SOMSD achieved 55% and 95% of accuracy respectively (no class information is provided for the quadtree dataset). We stress again, that such a procedure makes sense only when the class organisation of the data correlates with the driving force behind topographic map formation. If for example, the classes of trees are organised along the lines that cannot be reasonably captured by HMTM modelling, there is simply no reason why the achieved classification accuracy of GMT-HMTM should be high. But low classification rate would just mean that our model-driven topographic map formation does not correlate with the particular class labelling scheme. In such cases one can simply switch to local noise models that are more correlated with the class labelling. Alternatively, one might say that he/she wanted to see topographically organised data representations driven by aspects captured by HMTM (or any other noise model employed) and stick with the obtained topographic maps, irrespective of the class labels. This is an unsupervised learning setting after all. Again, without knowing the exact mechanism behind the topographic map formation, it is problematic to assign any performance-related interpretation to the classification rate obtained on the trained map.

Another advantage of a probabilistic model formulation is the possibility to inspect the tendency of the model to overfit the training data, by measuring the log-likelihood on an independent validation set. For example, for the TPB task, the validation set consists of 84 patterns produced in the same manner as the training set. During training, the log-likelihoods of the model on the training and validation sets were calculated in each iteration. The evolution of the log-likelihood for both data sets is presented in Fig. 4.14. It is apparent that the constrained nature of our model prevents it from overfitting the training sample.

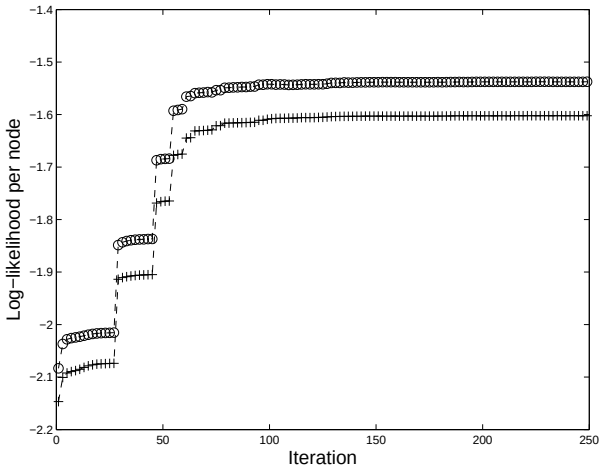


Fig. 4.14: Evolution of log-likelihood for GTM-HMTM on training (line with + marker) and validation (line with o marker) set for TPB.

4.4 Hidden Markov Models as Noise Models for GTM

In [Tiňo et al., 2004] the GTM is extended to the processing of data expressed as sequences. To that purpose, instead of formulating a constrained mixture of Gaussians, the technique relies on the formulation of a constrained mixture of HMMs. HMMs and their training procedure were reviewed in section 3.3. An HMM can be considered as a special case of HMTMs where the number of children is restricted to one. In this light, the GTM extension for sequences can be subsumed by the GTM-HMTM extension. Essentially all the machinery presented in section 4.2, is common to both approaches, with minor differences. In [Tiňo et al., 2004] the technique was demonstrated on sequences representing the web navigation of users and on melodic lines.

4.5 Markov Tree Models as Noise Models for GTM

4.5.1 Model Formulation

In this section we present an alternative extension of the GTM for the visualisation of R-regular tree-structured data, which employs MTMs (see 3.4.4) as noise models. Hence, we call this extension GTM-MTM. We follow the same methodology used formulating the GTM-HMTM extension in section 4.2. Since the methodology is common, our exposition covers only the key points.

A latent space $\mathcal{V} = [-1, +1]^2$ is defined and discretized, for the same aforementioned tractability reasons as in GTM and GTM-HMTM, by a rectangular grid of points $\boldsymbol{x}_c$, $c =$

$1, \ldots, C$. A RBF network maps each $\boldsymbol{x}_c$ to a set of MTM parameters $\Gamma(\boldsymbol{x}_c)$:

$$\Gamma(\boldsymbol{x}_c) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x}_c),$$

where $\boldsymbol{\phi}(\cdot) = [\phi_1(\cdot) \ldots \phi_M(\cdot)]^T$ and $\boldsymbol{W}$ is the weight matrix. Thus, each point $\boldsymbol{x}_c$ is mapped to a set of MTM parameters (namely to R transition matrices $\boldsymbol{B}^r(\boldsymbol{x}_c), r = 1, \ldots, R$) that address a MTM. Similarly to GTM and GTM-HMTM, the addressed MTMs belong to a constrained two-dimensional manifold $\mathcal{M}$ that is embedded in the space $\mathcal{H}$ of all MTMs of the same form as the ones induced by $\Gamma(\boldsymbol{x})$.

The number of MTM parameters is $R \times K \times K$, which means that $\boldsymbol{W}$ must be a matrix of dimensions $(R \times K \times K) \times M$ in order to generate the necessary number of parameters. However, it is more convenient to define $R \times K$ number of weight matrices instead of a single $\boldsymbol{W}$ matrix. This is possible since each parameter is generated independently of the others in mapping Γ. Thus, we define $R \times K$ number of weight matrices $\boldsymbol{W}^{r,k}$ of dimensions $K \times M$ for $r = 1, \ldots, R$ and $k = 1, \ldots, K$. We summarise the set of all such matrices in $\boldsymbol{\mathcal{W}} = \{\boldsymbol{W}^{r,k}\}_{r=1,\ldots,R,k=1,\ldots,K}$. Each matrix $\boldsymbol{W}^{r,k}$ generates the transition probabilities for the r-th child, from state k to all other K states:

$$\boldsymbol{B}^r(\boldsymbol{x}_c) = \{\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x})\}_{k=1,\ldots,K}. \tag{4.37}$$

However, the elements in $\boldsymbol{B}^r(\boldsymbol{x}_c)$ are probabilities, and must be in the range of $[0, 1]$, which is an issue since the RBF output in (4.37) is unbounded. Moreover, the sum of transition probabilities from a state k to all other states K must equal 1. The same problem was encountered in 4.2 when generating the initial probabilities and transition matrices for the GTM-HMTM. The same remedy is used here, namely applying softmax function g to the RBF output:

$$\boldsymbol{B}^r(\boldsymbol{x}_c) = \{g_l(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\}_{k,l=1,\ldots,K}, \tag{4.38}$$

We assume a dataset of independently generated regular trees $\mathcal{D} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$. The C latent points $\boldsymbol{x} \in \mathcal{V}$ are mapped via Γ to MTM models and form a mixture of C components.

Using (3.95), the model likelihood of this mixture is:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{\mathcal{W}}) &= \prod_{n=1}^{N}\sum_{c=1}^{C} P(\boldsymbol{x}_c)p(\boldsymbol{t}^{(n)}|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) = \prod_{n=1}^{N}\sum_{c=1}^{C} P(\boldsymbol{x}_c)\prod_{u=2}^{U_n} p(\boldsymbol{o}_u|\boldsymbol{o}_{\rho(u)},pos(u),\boldsymbol{\mathcal{W}}) \\ &= \frac{1}{C}\prod_{n=1}^{N}\sum_{c=1}^{C}\prod_{u=2}^{U_n} p(\boldsymbol{o}_u|\boldsymbol{o}_{\rho(u)},pos(u),\boldsymbol{\mathcal{W}}) \\ &\propto \prod_{n=1}^{N}\sum_{c=1}^{C}\prod_{u=2}^{U_n} p(\boldsymbol{o}_u|\boldsymbol{o}_{\rho(u)},pos(u),\boldsymbol{\mathcal{W}}), \end{aligned} \tag{4.39}$$

where we discard constant $\frac{1}{C}$. We take the logarithm and use (3.96), so that the (scaled) log-likelihood reads:

$$\begin{aligned}\log\mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D}) &\propto \sum_{n=1}^{N}\log\sum_{c=1}^{C}\prod_{u=2}^{U_n} p(\boldsymbol{o}_u|\boldsymbol{o}_{\rho(u)},pos(u),\boldsymbol{\mathcal{W}}) \\ &\propto \sum_{n=1}^{N}\log\sum_{c=1}^{C}\prod_{u=2}^{U_n}\prod_{k=1}^{K}\prod_{l=1}^{K}\prod_{r=1}^{R} p(\boldsymbol{o}_u=l|\boldsymbol{o}_{\rho(u)}=k,pos(u)=r,\boldsymbol{\mathcal{W}})^{\delta_{\boldsymbol{o}_{\rho(u)},k}\delta_{\boldsymbol{o}_u,l}\delta_{pos(u),r}}. \end{aligned} \tag{4.40}$$

4.5.2 Model training

We perform training of the GTM-MTM via the EM algorithm. Similarly to section 3.2, we postulate the following hidden variables that indicate the unobserved component origin for each data item:

$$z_c^{(n)} = \begin{cases} 1, & \text{component } c \text{ generated the tree } \boldsymbol{y}^{(n)}; \\ 0, & \text{otherwise.} \end{cases} \tag{4.41}$$

The conditional expectation of the hidden variables is calculated in the E-step by the following posterior probabilities:

$$E[z_c^{(n)}|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}] = p(z_c^{(n)}=1|\boldsymbol{\mathcal{W}}^{(i)}) = p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\Theta}^{(i)}) = \frac{P(\boldsymbol{x}_c)p(\boldsymbol{y}^{(n)}|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})}{\sum_{c'=1}^{C} P(\boldsymbol{x}_{c'})p(\boldsymbol{y}^{(n)}|\boldsymbol{x}_{c'},\boldsymbol{\mathcal{W}}^{(i)})}. \tag{4.42}$$

Having calculated the conditional expectation of the hidden variables, we proceed with the calculation of the expected complete-data log-likelihood, using (3.97):

$$\begin{aligned}&E_{\mathcal{Z}}[\log\mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}] \propto \\ &\sum_{n=1}^{N}\sum_{c=1}^{C}\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{r=1}^{R} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)})\nu_{rkl}^{(n)}\log p(\boldsymbol{o}_u=l|\boldsymbol{o}_{\rho(u)}=k,pos(u)=r,\boldsymbol{\mathcal{W}}). \end{aligned} \tag{4.43}$$

In the M-step we take derivatives of (4.43) with respect to j, m element w_{jm}^{st} of matrix $\boldsymbol{W}^{s,t}$:

$$\begin{aligned}
\frac{\partial}{\partial w_{jm}^{st}} E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}] &\propto \frac{\partial}{\partial w_{jm}^{st}} \sum_{n=1}^{N}\sum_{c=1}^{C}\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{r=1}^{R} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)})\nu_{rkl}^{(n)} \\
&\times \log p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}) \\
&\propto \sum_{n=1}^{N}\sum_{c=1}^{C}\sum_{l=1}^{K} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)})\nu_{stl}^{(n)} \frac{\partial}{\partial w_{jm}^{st}} \log p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}}) \\
&\propto \sum_{n=1}^{N}\sum_{c=1}^{C}\sum_{l=1}^{K} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)})\nu_{stl}^{(n)} \frac{1}{p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}})} \\
&\times \frac{\partial}{\partial w_{jm}^{st}} p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}}).
\end{aligned} \tag{4.44}$$

Derivative $\frac{\partial}{\partial w_{jm}^{st}} p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}})$, calculated in Appendix C, is equal to:

$$\begin{aligned}
\frac{\partial}{\partial w_{jm}^{st}} p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}}) &= \frac{\partial}{\partial w_{jm}^{st}} \frac{\exp(\boldsymbol{W}_{l}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))}{\sum_{l'=1}^{K}\exp(\boldsymbol{W}_{l'}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))} \\
&= \phi_m(\boldsymbol{x}_c) p(\boldsymbol{o}_u = j|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}}) \\
&\times \Big(\delta_{l,j} - p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}})\Big).
\end{aligned} \tag{4.45}$$

Substitute (4.45) in (4.44) to obtain the equation:

$$\begin{aligned}
\frac{\partial}{\partial w_{jm}^{st}} E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}] \propto \sum_{n=1}^{N}\sum_{c=1}^{C}\sum_{l=1}^{K} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)})\nu_{stl}^{(n)} \frac{1}{p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}})} \\
\times \phi_m(\boldsymbol{x}_c) p(\boldsymbol{o}_u = j|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}}) \\
\times \Big(\delta_{l,j} - p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}})\Big).
\end{aligned} \tag{4.46}$$

Thus, we finally obtain the derivative in (4.46) that can be used for optimising the model.

4.6 Experimental Results for GTM-MTM

We experimented with two datasets, a toy dataset constructed by sampling three MTM models, and the quadtree dataset that was used in GTM-HMTM.

The complexity of the algorithm in the E-step is $\mathcal{O}(CNRK^2)$, which follows from the number

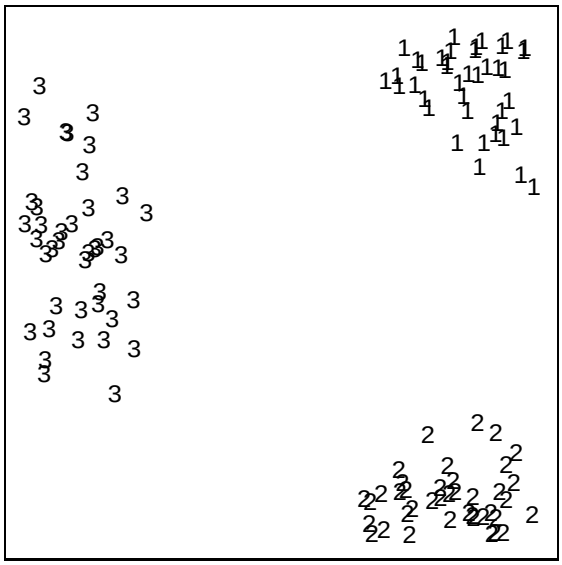


Fig. 4.15: Visualisation of toy dataset for GTM-MTM.

of operations needed in (3.97) to calculate the log-likelihood of a single data item, multiplied by the number C of latent points and the number N of data items in the dataset. Regarding the M-step, the complexity is that of the scaled conjugate gradient algorithm [Moller, 1993] which is $\mathcal{O}(2W^2)$, where W is the number of free parameters to be optimised. For the GTM-MTM the number of parameters is $W = MRK^2$ which is the number of elements in matrix $\boldsymbol{\mathcal{W}}$. A MATLAB implementation of GTM-MTM (on the same machine as in 4.3.2) required 1 hour for the toy dataset and 6 hours for the quadtree dataset to learn a satisfactory topographic map.

For the toy dataset we set $K = 2$ and instantiated 3 MTMs with randomly generated parameters to simulate three classes of trees. The results for the toy dataset are displayed in Fig. 4.15. The three clusters corresponding to the three randomly instantiated MTMs are clearly discerned. Data points are numbered to indicate the MTM they originate from. Regarding the quadtree dataset, the results are displayed in Fig. 4.16. Training GTM-MTM with the original resolution of 64×64 did not yield any numerical problems as experienced in GTM-MTM. Clearly, GTM-MTM has achieved a much higher quality of topographic organisation than GTM-HMTM. The upper-left corner is dominated by images of ducks facing to the right, while the lower-right corner is dominated by images facing to the left. In between the two, close to the centre of the map, we find images of frontal views. Finally, as we move from the lower left corner towards the top, we come across images of rear views.

We also examined the tendency of GTM-MTM to overfit the training set. For the case of the quadtree dataset, we show log-likelihood evolutions in Fig. 4.17(a) and 4.17(b) on the training and validation set. We examine two training sessions. The first one corresponds to a training

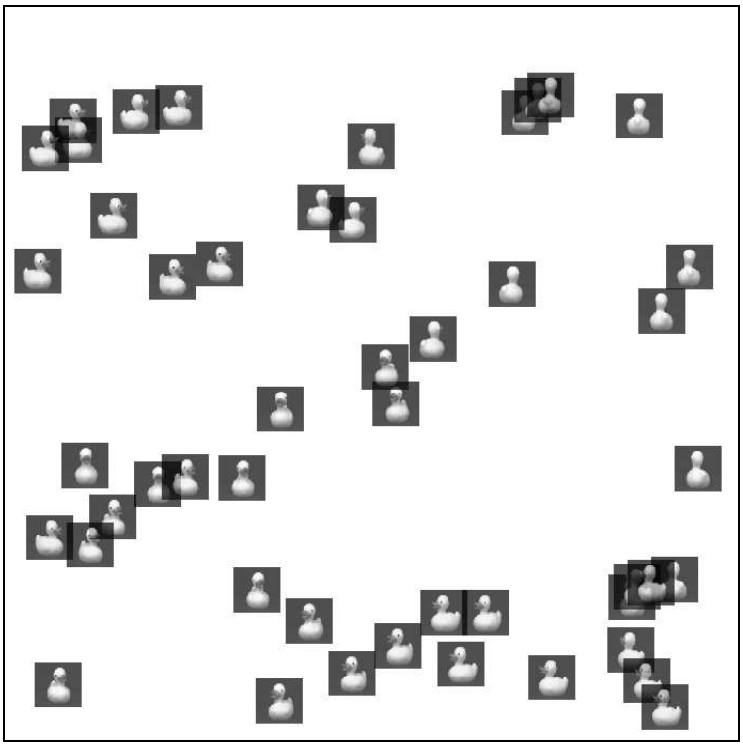
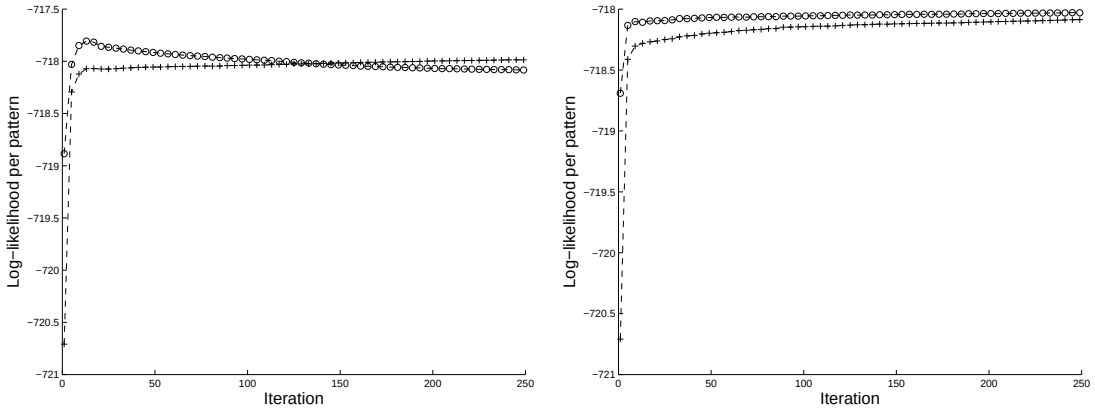


Fig. 4.16: Visualisation of quadtree dataset (64×64) for GTM-MTM.

session where overfitting occurs (the topographic map of this session is the one displayed in 4.16). In order to avoid overfitting we changed the variance of the radial basis functions ϕ of the RBF network from 1.0 to 2.0 and performed a second training session. Increasing the variance in the RBF network, has the effect of making the basis functions wider, less localised, blending them to a higher degree in their overlapping regions. In terms of the GTM, this has the effect of not allowing local noise models to become very different from their neighbours in terms of parameters, which enforces a form of regularisation. The evolution of the log-likelihood with RBF variance equal to 2.0 is illustrated in Fig. 4.17(b). Even though, we observed practically identical topographic organisations in both training sessions, the second session achieves better generalisation performance, which is advantageous if new data items are to be projected on the map. We stress, that checking the model likelihood on a hold-out sample is a natural way of detecting possible overfitting. In case of a highly overfitted model, it would be difficult to interpret visualisation plots as representing any general tendency in the data.



(a) (b)

Fig. 4.17: Evolution of log-likelihood for GTM-MTM on training (line with + marker) and validation (line with o marker) set. In (a) the variance of the radial basis functions is set to 1.0 and in (b) it is set to 2.0.

Chapter 5

Magnification Factors for Topographic Mapping

In chapter 4 we saw that topographic organisation of a dataset on a two-dimensional latent space enables us to visualise the data points and infer potential clusters and relationships between them, by interpreting the distances between them. However, these distances can be "deceiving" without proper interpretation; one must keep in mind that mapping Γ from the latent space $\mathcal{V}$ to the manifold $\mathcal{M}$ of local noise models is non-linear. Even though mapping Γ retains the local neighbourhood of the projection of each point, in general the distances between projections are not preserved. We illustrate this with a simple analogy. Consider the mapping of points x that belong on a line segment $[-2, 2]$ via $f(x) = -x^2$ to y values, as depicted in Fig. 5.1. Points $x = 1$ and $x = 0$ are equidistant from $x = 0.5$. However, this does not apply for their respective images, as $y = -1, y = 0$ are not equidistant from $y = -0.25$. This incurred expansion or magnification is not visible on the line segment. Another possibility is the occurrence of a contraction in the mapping. Furthermore in more complex mappings both effects may be manifested in various degrees. Similarly in GTM and its extensions, RBF function Γ ((4.2) for GTM, (4.17)-(4.19) for GTM-HMTM and (4.38) for GTM-MTM) is a nonlinear function. Thus on the latent map, two data items that lie close to each other may in fact be separated as the space between them is magnified. These magnifications are not visible in the latent space.

In order to appreciate how the latent space is magnified we present two approaches for measuring the magnification around latent points $\boldsymbol{x} \in \mathcal{V}$. One approach relies on the local approximation of the *Kullback-Leibler divergence* (KLD). Each latent point $\boldsymbol{x}$ is perturbed by an infinitesimal distance of $\boldsymbol{dx}$ in various directions. If $p(\cdot|\boldsymbol{x})$ is the noise model that corresponds to $\boldsymbol{x}$, then $p(\cdot|\boldsymbol{x} + \boldsymbol{dx})$ is the noise model addressed by the perturbed version $\boldsymbol{x} + \boldsymbol{dx}$ of $\boldsymbol{x}$. This

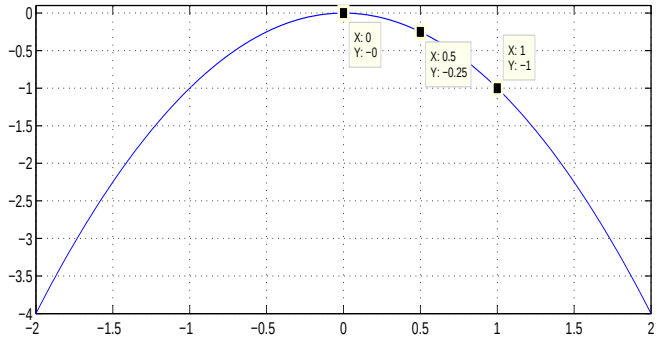


Fig. 5.1: Graph of $f(x) = -x^2$.

situation is illustrated in Fig. 5.2. For small perturbations we expect the distribution of the perturbed noise model $p(\cdot|\boldsymbol{x} + \boldsymbol{dx})$ to closely resemble the original distribution $p(\cdot|\boldsymbol{x})$. However, perturbations corresponding to different directions will result to different noise models, some more distant (statistically) to $p(\cdot|\boldsymbol{x})$ than others; how distant is measured via the KLD.

The second approach relies on the *Fisher information matrix* (FIM). This concept makes explicit use of the *geometry* of the models in space $\mathcal{M}$. Models are infinitesimally perturbed and the distance between the original model $p(\cdot|\boldsymbol{x})$ and its perturbed version $p(\cdot|\boldsymbol{x} + \boldsymbol{dx})$ is measured via the Fisher information matrix that acts as a metric tensor on the induced Riemannian manifold [Kullback, 1959].

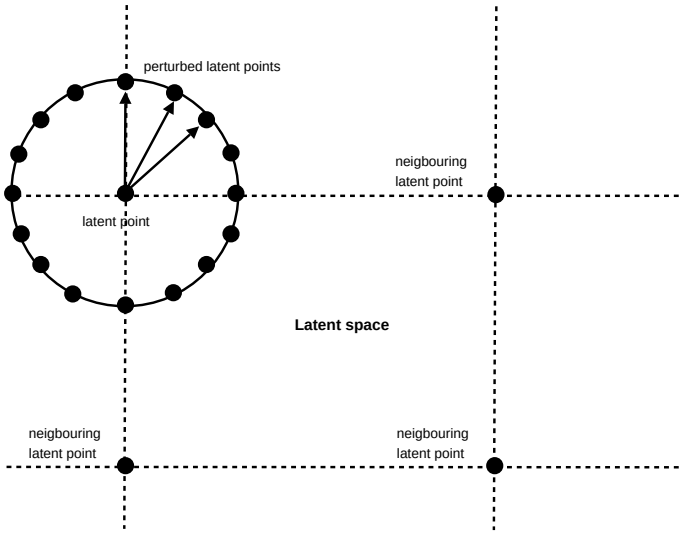


Fig. 5.2: Magnification factors may be measured via the perturbation of a latent point in regular intervals on a small circle.

We shall first discuss how magnification factors are obtained in the original GTM algorithm. We then present our own two approaches of measuring magnification factors, via KLD and

FIM, and calculate them for GTM-HMM, GTM-HMTM and GTM-MTM. We also re-derive magnification factors for GTM using these approaches and arrive at similar results.

5.1 Magnification Factors for Original GTM

Magnification factors for the GTM are presented in [Bishop et al., 1997]. Here however, we follow a geometrical line of thought presented in [Svensén, 1998]. Each point $\boldsymbol{x}$ of latent space $\mathcal{V}$ is mapped via $\Gamma(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x})$ (see (4.2)), implemented as a RBF network, to the high-dimensional space $\mathbb{R}^d$ where the vector data items $\boldsymbol{t}$ reside. The Jacobian of mapping Γ is the $M \times d$ matrix:

$$\boldsymbol{J} = \begin{pmatrix} \boldsymbol{W}_1 \frac{\partial}{\partial x_1}\boldsymbol{\phi}(\boldsymbol{x}) & \dots & \boldsymbol{W}_1 \frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x}) \\ \boldsymbol{W}_2 \frac{\partial}{\partial x_1}\boldsymbol{\phi}(\boldsymbol{x}) & \dots & \boldsymbol{W}_2 \frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x}) \\ \vdots & \vdots & \vdots \\ \boldsymbol{W}_d \frac{\partial}{\partial x_1}\boldsymbol{\phi}(\boldsymbol{x}) & \dots & \boldsymbol{W}_d \frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x}) \end{pmatrix}, \tag{5.1}$$

or in a more compact form:

$$\boldsymbol{J} = \begin{pmatrix} \boldsymbol{W}\frac{\partial}{\partial x_1}\boldsymbol{\phi}(\boldsymbol{x}) & \dots & \boldsymbol{W}\frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x}) \end{pmatrix}, \tag{5.2}$$

where $r = 1, \dots, q$ with q the dimension of the latent space.

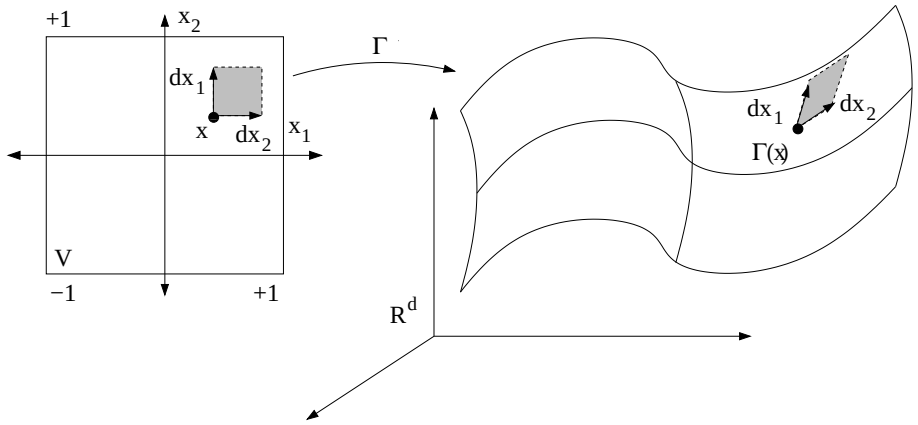


Fig. 5.3: An area element in the latent space when mapped to the high-dimensional space is subject to magnification.

The Jacobian relates displacements $\boldsymbol{dx}$ in $\mathcal{V}$ to displacements $\boldsymbol{dy}$ in space $\mathbb{R}^d$ by $\boldsymbol{dy} = \boldsymbol{J}\boldsymbol{dx}$. As illustrated in Fig. 5.3, we can take two very small orthogonal displacements $\boldsymbol{dx}_1$ and $\boldsymbol{dx}_2$ of equal length $\|\boldsymbol{dx}_1\| = \|\boldsymbol{dx}_2\| = x$, along the axis in $\mathcal{V}$, which correspond to displacements

$\boldsymbol{dy}_1$ and $\boldsymbol{dy}_2$ in $\mathbb{R}^d$ respectively. The area defined by $\boldsymbol{dx}_1$ and $\boldsymbol{dx}_2$ is transformed to the area defined by $\boldsymbol{dy}_1$ and $\boldsymbol{dy}_2$. This is illustrated in Fig. 5.3. A result in [Bloom, 1979] states that the (hyper)volume of the parallelepiped defined by the rows of a matrix $\boldsymbol{A}$ is equal to $\det(\boldsymbol{A}^T\boldsymbol{A})^{\frac{1}{2}}$. We form matrices $\boldsymbol{D_x} = \left[\begin{array}{cc} \boldsymbol{dx}_1 & \boldsymbol{dx}_2 \end{array}\right]$ and $\boldsymbol{D_y} = \left[\begin{array}{cc} \boldsymbol{dy}_1 & \boldsymbol{dy}_2 \end{array}\right]$ of dimensions $d \times d$ and $M \times d$ respectively. We note that $\boldsymbol{D_x} = xI_d$. We obtain:

$$\begin{aligned} \boldsymbol{D_y} &= \boldsymbol{J}\boldsymbol{D_x}, \\ \boldsymbol{D_y}^T\boldsymbol{D_y} &= \boldsymbol{D_x}^T\boldsymbol{J}^T\boldsymbol{J}\boldsymbol{D_x}, \\ \det(\boldsymbol{D_y}^T\boldsymbol{D_y}) &= \det(\boldsymbol{D_x}^T\boldsymbol{J}^T\boldsymbol{J}\boldsymbol{D_x}) = \det(x\boldsymbol{J}^T\boldsymbol{J}x) = x^2\det(\boldsymbol{J}^T\boldsymbol{J}), \end{aligned} \tag{5.3}$$

where

$$\begin{aligned} \boldsymbol{J}^T\boldsymbol{J} &= \left(\frac{\partial}{\partial x_1}\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{W}^T\boldsymbol{W}\frac{\partial}{\partial x_1}\boldsymbol{\phi}(\boldsymbol{x}) \quad \ldots \quad \frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{W}^T\boldsymbol{W}\frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x})\right) \\ &= \boldsymbol{\gamma}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{\gamma}, \end{aligned} \tag{5.4}$$

where vector $\boldsymbol{\gamma}$ stores all partial derivatives $\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}),\ r = 1, \ldots, q$. Hence, the ratio of the two squared areas is equal to :

$$\frac{\det(\boldsymbol{D_y}^T\boldsymbol{D_y})}{\det(\boldsymbol{D_x}^T\boldsymbol{D_x})} = \frac{\det(\boldsymbol{D_y}^T\boldsymbol{D_y})}{x^2} = \det(\boldsymbol{\gamma}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{\gamma}), \tag{5.5}$$

with $x^2 = 1$ if the columns in $\boldsymbol{D_x}$ form an orthonormal basis.

A ratio higher than 1 signifies that the area around latent point $\boldsymbol{x}$ is magnified under mapping Γ, which means that the underlying local noise models, here Gaussian densities, vary in a statistical sense from their neighbours. On the contrary, a ratio lower than 1 means that the area is contracted under the mapping and that the underlying local Gaussians are very similar to each other.

5.2 Kullback-Leibler Divergence

The Kullback-Leibler divergence (KLD) or relative entropy is a scalar quantity that informs us of the "distance" between two distributions P and Q (assuming that $P(\boldsymbol{t}) > 0, Q(\boldsymbol{t}) > 0, \forall \boldsymbol{t} \in \mathcal{T}$) and is defined as [Cover and Thomas, 1991]:

$$D_{KL}[P||Q] = \int_{\mathcal{T}} P(\boldsymbol{t}) \log \frac{P(\boldsymbol{t})}{Q(\boldsymbol{t})} \boldsymbol{dt}. \tag{5.6}$$

and for the discrete case:

$$\kappa[\boldsymbol{w}||\boldsymbol{w}'] = \sum_i w_i \log \frac{w_i}{w'_i}. \tag{5.7}$$

An important property, known as Gibbs' inequality, is that KLD is always non-negative with a minimum of zero when Q exactly matches distribution P, $D_{KL}[P||P] = 0$ [Cover and Thomas, 1991]. It is important to note that KLD is not symmetric in general $D_{KL}[P||Q] \neq D_{KL}[Q||P]$, hence it does not constitute a proper distance metric. By writing KLD as:

$$D_{KL}[P||Q] = \int_{\mathcal{T}} P(\boldsymbol{t}) \log P(\boldsymbol{t}) d\boldsymbol{t} - \int_{\mathcal{T}} P(\boldsymbol{t}) \log Q(\boldsymbol{t}) d\boldsymbol{t}, \tag{5.8}$$

it may be interpreted as the *statistical distance* of approximating distribution P with distribution Q. In [Rabiner, 1989] it is noted that KLD acts as a statistical measure between HMMs and can be understood as a measure of how well HMM $p(\cdot|\boldsymbol{\theta})$ matches observations generated by model $p(\cdot|\boldsymbol{\theta}')$ relative to how well HMM $p(\cdot|\boldsymbol{\theta})$ matched the observations generated by itself. In general, KLD can be applied to all kinds of probabilistic models. The models do not even need to be both of the same form, e.g. we can measure the distance between a Gaussian and a binomial distribution.

In practice, for two models $p(\cdot|\boldsymbol{\theta})$ and $p(\cdot|\boldsymbol{\theta}')$, KLD can be measured as the observed $\hat{D}_{KL}$ via a Monte-Carlo type approximation:

$$\hat{D}_{KL}[p(\cdot|\boldsymbol{\theta})||p(\cdot|\boldsymbol{\theta}')] = \sum_{\boldsymbol{t}\in\mathcal{T}} p(\boldsymbol{t}|\boldsymbol{\theta}) \log p(\boldsymbol{t}|\boldsymbol{\theta}) - \sum_{\boldsymbol{t}\in\mathcal{T}} p(\boldsymbol{t}|\boldsymbol{\theta}) \log p(\boldsymbol{t}|\boldsymbol{\theta}'). \tag{5.9}$$

Even though the above formula is a robust method of estimating KLD and applicable for all practical settings, we can do better than that and approximate KLD with efficient closed-form formulae. In the following sections we demonstrate how KLD can be measured for the density models presented in chapter 3, namely Gaussian densities, HMMs, HMTMs, HMMs and mixture models. In particular, the calculation of KLD for Gaussian densities leads to the same magnification factors calculated for GTM in section 5.1.

5.2.1 KLD for Gaussian Densities

For two d-dimensional Gaussian distributions $P(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $Q(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}; \boldsymbol{\mu}', \boldsymbol{\Sigma}')$ the formula for KLD is [Kullback, 1959]:

$$\begin{aligned} D_{KL}[P||Q] &= \int_{\mathcal{T}} P(\boldsymbol{t}) \log \frac{P(\boldsymbol{t})}{Q(\boldsymbol{t})} d\boldsymbol{t} \\ &= \int_{\mathcal{T}} \mathcal{N}(\boldsymbol{t}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \log \frac{\mathcal{N}(\boldsymbol{t}; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\mathcal{N}(\boldsymbol{t}; \boldsymbol{\mu}', \boldsymbol{\Sigma}')} d\boldsymbol{t} \\ &= \frac{1}{2}\left[\log \frac{\det \boldsymbol{\Sigma}'}{\det \boldsymbol{\Sigma}} - d + tr(\boldsymbol{\Sigma}'^{-1}\boldsymbol{\Sigma}) + (\boldsymbol{\mu} - \boldsymbol{\mu}')^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \boldsymbol{\mu}')\right]. \end{aligned} \tag{5.10}$$

The formula circumvents the need of a Monte-Carlo approximation and calculates the precise KLD for two Gaussian distributions.

In the particular case of GTM, where the local noise models are implemented by spherical Gaussians of the same variance σ^2, the KLD between two neighbouring models $p(\cdot|\boldsymbol{x})$ and $p(\cdot|\boldsymbol{x} + \boldsymbol{dx})$ is:

$$\begin{aligned} D_{KL}[p(\cdot|\boldsymbol{x})||p(\cdot|\boldsymbol{x} + \boldsymbol{dx})] &= \frac{1}{2\sigma^2}(\boldsymbol{\mu} - \boldsymbol{\mu}')^T(\boldsymbol{\mu} - \boldsymbol{\mu}') \\ &= \frac{1}{2\sigma^2}(\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}(\boldsymbol{x} + \boldsymbol{dx}))^T \boldsymbol{W}^T \boldsymbol{W}(\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}(\boldsymbol{x} + \boldsymbol{dx})) \\ &= \frac{1}{2\sigma^2} \boldsymbol{d\phi}(\boldsymbol{x})^T \boldsymbol{W}^T \boldsymbol{W} \boldsymbol{d\phi}(\boldsymbol{x}), \end{aligned} \tag{5.11}$$

where the primed quantities correspond to model $p(\cdot|\boldsymbol{x} + \boldsymbol{dx})$. This formula is in correspondence (ignoring constants) to the magnification factors of GTM in (5.5) (where the ratio of areas was considered).

5.2.2 KLD for Mixture Models

Before moving to the approximation of KLD for HMMs and HMTMs, we present a result that holds for two general mixtures of C components [Singer and Warmuth, 1998]:

$$P(\boldsymbol{t}) = \sum_{c=1}^{C} P(c)p(\boldsymbol{t}|c), \quad Q(\boldsymbol{t}) = \sum_{c=1}^{C} Q(c)q(\boldsymbol{t}|c),$$

where $P(c)$ is subject to the constraints $0 \leq P(c) \leq 1$ and $\sum_{c=1}^{C} P(c) = 1$ and so is $Q(c)$. Using the log-sum inequality [Cover and Thomas, 1991], an upper bound D can be derived for the

KLD:

$$
\begin{aligned}
D_{KL}[P||Q] &= \int_{\mathcal{T}} P(\boldsymbol{t}) \log \frac{P(\boldsymbol{t})}{Q(\boldsymbol{t})} \boldsymbol{dt} = \int_{\mathcal{T}} \sum_{c=1}^{C} p(c)p(\boldsymbol{t}|c) \log \frac{\sum_{c=1}^{M} P(c)p(\boldsymbol{t}|c)}{\sum_{c=1}^{M} Q(c)q(\boldsymbol{t}|c)} \boldsymbol{dt} \\
&\leq \int_{\mathcal{T}} \sum_{c=1}^{C} P(c)p(\boldsymbol{t}|c) \log \frac{P(c)p(\boldsymbol{t}|c)}{Q(c)q(\boldsymbol{t}|c)} \boldsymbol{dt}, \\
D[P||Q] &= \sum_{c=1}^{C} P(c) \log \frac{P(c)}{Q(c)} + \sum_{c=1}^{C} P(c) \int_{\mathcal{T}} p(\boldsymbol{t}|c) \log \frac{p(\boldsymbol{t}|c)}{q(\boldsymbol{t}|c)} d\boldsymbol{t} \\
D[P||Q] &= \kappa[(P(1) \dots P(c))^T || (Q(1) \dots Q(c))^T] + \sum_{c=1}^{C} P(c) D_{KL}[p(\boldsymbol{t}|c)||q(\boldsymbol{t}|c)]. \quad (5.12)
\end{aligned}
$$

Thus, the KLD between two mixtures is approximated by an upper bound $D[P||Q]$.

This result can be readily specialised for a mixture of Gaussians by substituting the components with Gaussians or further extended for mixtures of Markov models with the aid of results presented in the following sections. Bound $D[P||Q]$ in conjunction with (5.10) estimates efficiently an upper limit on the statistical divergence of Gaussian mixtures.

5.2.3 KLD for Hidden Markov Tree Models and Hidden Markov Models

In [Do, 2003] an efficient procedure is presented that estimates an upper bound for KLD in HMTMs and as a special case in HMMs. The procedure relies on the upward-recursion described by (3.77)-(3.79). At each iteration it estimates an upper bound D_k, for state $k = 1 \dots K$, for the KLD between the partially calculated upward probabilities for two HMTMs. Specifically, the KLD between two HMTMs $p(\cdot|\boldsymbol{\theta})$ and $p(\cdot|\boldsymbol{\theta}')$ can be approximated as follows:

- Recursion initiates at leaf nodes; emissions $p(\boldsymbol{o}_u|q_u = k, \boldsymbol{\theta})$ are Gaussian densities and we apply (5.10) to (3.77):

$$
D_k[u; \boldsymbol{\theta}, \boldsymbol{\theta}'] = D_{KL}[p(\boldsymbol{o}_u|q_u = k, \boldsymbol{\theta})||p(\boldsymbol{o}_u|q_u = k, \boldsymbol{\theta}')] = D_{KL}[\mathcal{N}(\boldsymbol{o}_u; \boldsymbol{\mu}_k, \sigma_k I))||\mathcal{N}(\boldsymbol{o}_u; \boldsymbol{\mu}'_k, \sigma'_k I)]. \quad (5.13)
$$

- Recursive step; apply (5.12) to equation (3.78):

$$\begin{aligned} D_{KL}[\beta_k(u;\boldsymbol{\theta})||\beta_k(u;\boldsymbol{\theta}')] &= D_{KL}[p(\boldsymbol{o}_u|q_u=k,\boldsymbol{\theta})||p(\boldsymbol{o}_u|q_u=k,\boldsymbol{\theta}')] \\ &\quad + \sum_{v\in ch(u)} D_{KL}\Big[\sum_{i=1}^{K}\beta_i(v;\boldsymbol{\theta})p(q_v=i|q_u=k,\boldsymbol{\theta})||\sum_{i=1}^{K}\beta_i(v;\boldsymbol{\theta}')p(q_v=i|q_u=k,\boldsymbol{\theta}')\Big], \\ D_{KL}[\beta_k(u;\boldsymbol{\theta})||\beta_k(u;\boldsymbol{\theta}')] &\leq D_k[u;\boldsymbol{\theta},\boldsymbol{\theta}'] = D_{KL}[\mathcal{N}(\boldsymbol{o}_u;\boldsymbol{\mu}_k,\sigma_k I))||\mathcal{N}(\boldsymbol{o}_u;\boldsymbol{\mu}'_k,\sigma'_k)] \\ &\quad + \sum_{v\in ch(u)} \Big(\kappa[p(q_v|q_u=k,\boldsymbol{\theta})||p(q_v|q_u=k,\boldsymbol{\theta}')] + \sum_{i=1}^{K} p(q_v=i|q_u=k,\boldsymbol{\theta})D_i[v;\boldsymbol{\theta},\boldsymbol{\theta}']\Big). \end{aligned} \tag{5.14}$$

- Final step; the upper bound of KLD is found by applying again (5.12) to equation (3.79):

$$\mathcal{K}[\boldsymbol{y};\boldsymbol{\theta},\boldsymbol{\theta}'] = \kappa[p(q_1|\boldsymbol{\theta})||p(q_1|\boldsymbol{\theta}')] + \sum_{k=1}^{K} p(q_1=k|\boldsymbol{\theta})D_k[1;\boldsymbol{\theta},\boldsymbol{\theta}']. \tag{5.15}$$

The above recursive estimation of an upper bound for KLD depends solely on the parameters of the models and is evidently more efficient than a Monte-Carlo approximation as no sampling of the involved distributions is necessary. This recursion can be readily specialised for HMMs by considering HMMs as a special case of HMTMs, by restricting the number of children of each node to one. Thus, a tree structure degenerates to a sequence (a tree where each node has no more than one child). Therefore, the KLD between two HMMs $p(\cdot|\boldsymbol{\theta})$ and $p(\cdot|\boldsymbol{\theta}')$, can be approximated as follows:

- Recursion starts at the end of the sequence; apply (5.10) to (3.49):

$$\begin{aligned} D_{KL}[\beta_k(T;\boldsymbol{\theta})||\beta_k(T;\boldsymbol{\theta}')] &= D_k[T;\boldsymbol{\theta},\boldsymbol{\theta}'] = D_{KL}[p(\boldsymbol{s}_T|q_T=k,\boldsymbol{\theta})||p(\boldsymbol{s}_T|q_T=k,\boldsymbol{\theta}')] \\ &= D_{KL}[\mathcal{N}(p(\boldsymbol{s}_T;\boldsymbol{\mu}_k,\sigma_k I))||\mathcal{N}(p(\boldsymbol{s}_T;\boldsymbol{\mu}'_k,\sigma'_k I)]. \end{aligned} \tag{5.16}$$

- Recursive step; apply (5.12) to equation (3.78):

$$
\begin{aligned}
&D_{KL}[\beta_k(t;\boldsymbol{\theta})||\beta_k(t;\boldsymbol{\theta}')] = \\
&\quad D_{KL}[p(\boldsymbol{s}_t|q_t=k,\boldsymbol{\theta})||p(\boldsymbol{s}_t|q_t=k,\boldsymbol{\theta}')] \\
&\quad +D_{KL}\Big[\sum_{i=1}^{K}\beta_i(t+1;\boldsymbol{\theta})p(q_{t+1}=i|q_t=k,\boldsymbol{\theta})||\sum_{i=1}^{K}\beta_i(t+1;\boldsymbol{\theta}')p(q_{t+1}=i|q_t=k,\boldsymbol{\theta}'))\Big], \\
&D_{KL}[\beta_k(t;\boldsymbol{\theta})||\beta_k(t;\boldsymbol{\theta}')] \leq D_k[u;\boldsymbol{\theta},\boldsymbol{\theta}'] = \qquad (5.17) \\
&\quad D_{KL}[\mathcal{N}(\boldsymbol{s}_t;\boldsymbol{\mu}_k,\sigma_k I))||\mathcal{N}(\boldsymbol{s}_t;\boldsymbol{\mu}'_k,\sigma'_k)] \\
&\quad +\kappa[p(q_{t+1}|q_t=k,\boldsymbol{\theta})||p(q_{t+1}|q_t=k,\boldsymbol{\theta}')]+\sum_{i=1}^{K}p(q_{t+1}=i|q_t=k,\boldsymbol{\theta})D_i[t+1;\boldsymbol{\theta},\boldsymbol{\theta}'].
\end{aligned}
\qquad (5.18)
$$

- Final step; the upper bound of KLD is found by applying again (5.12) to equation (3.79):

$$\mathcal{K}[\boldsymbol{y};\boldsymbol{\theta},\boldsymbol{\theta}'] = \kappa[p(q_1|\boldsymbol{\theta})||p(q_1|\boldsymbol{\theta}')]+\sum_{k=1}^{K}p(q_1=k|\boldsymbol{\theta})D_k[1;\boldsymbol{\theta},\boldsymbol{\theta}']. \qquad (5.19)$$

Thus the KLD between two HMTs or HMTMs is approximated by the upper bound as:

$$D_{KL}[p(\cdot|\boldsymbol{\theta})||(p(\cdot|\boldsymbol{\theta}')] \leq \mathcal{K}[\boldsymbol{y};\boldsymbol{\theta},\boldsymbol{\theta}']. \qquad (5.20)$$

5.2.4 KLD for Markov Tree Models

For MTMs we can resort to a fast approximation of KLD by assuming that generated trees are sufficiently deep and that the compared MTMs are not too dissimilar. Using a result from [Falkhausen et al., 1995] for HMMs, we approximate the KLD between a MTM addressed by $p(\cdot|\boldsymbol{\theta})$ and its perturbed version $p(\cdot|\boldsymbol{\theta}+\boldsymbol{d\theta})$ by:

$$
\begin{aligned}
\hat{D}_{KL}[p(\cdot|\boldsymbol{\theta})||p(\cdot|\boldsymbol{\theta}+\boldsymbol{d\theta})] &= \sum_{r=1}^{R}\sum_{k=1}^{K}\pi_k^r\sum_{l=1}^{K}p(\boldsymbol{o}_u=l|\boldsymbol{o}_{\rho(u)}=k,pos(u)=r|\boldsymbol{\theta}) \\
&\times \log\frac{p(\boldsymbol{o}_u=l|\boldsymbol{o}_{\rho(u)}=k,pos(u)=r|\boldsymbol{\theta})}{p(\boldsymbol{o}_u=l|\boldsymbol{o}_{\rho(u)}=k,pos(u)=r|\boldsymbol{\theta}+\boldsymbol{d\theta})},
\end{aligned}
\qquad (5.21)
$$

where probabilities π_k^r are obtained as the normalised left eigenvector when solving $\boldsymbol{\pi}^r=\boldsymbol{\pi}^r\boldsymbol{B}^{(r)}$.

5.2.5 KLD as a Magnification Factor for the GTM Extensions

In the preceding sections of 5.2.3 and 5.2.4 we approximate the KLD for HMTMs, HMMs and MTMs, without considering them as part of a GTM-type parametrisation, i.e. parameter vectors $\boldsymbol{\theta}$ are free parameters, not generated via some constrained parametrisation. However, we can also approximate KLD between noise models that belong on the manifold $\mathcal{M}$ induced by a GTM formulation using the same preceding equations. Noise models in GTM are parametrised by latent points $\boldsymbol{x}$, thus $\boldsymbol{x}$ plays the role of parameter vector $\boldsymbol{\theta}$. As aforementioned in the introduction of this chapter, magnification factors can be measured by perturbing a point $\boldsymbol{x}$ of the latent space by an infinitesimal displacement $\boldsymbol{dx}$ in various directions. Point $\boldsymbol{x}$ addresses a model $p(\cdot|\boldsymbol{x})$ on the manifold and its displaced version $\boldsymbol{x} + \boldsymbol{dx}$ addresses a model $p(\cdot|\boldsymbol{x} + \boldsymbol{dx})$. The statistical distance between these two models is estimated via the KLD approximations derived in the preceding sections of 5.2.3 and 5.2.4. This perturbation scheme is illustrated in Fig. 5.2. Of course, we could also measure the KLD between two models on the manifold that are not close neighbours, but belong to regions of the manifold distant from each other, in order to reveal other properties of the map apart from local magnification factors. Even though KLD would yield a valid result, that would be irrespective of the geometry of the manifold. A correct approach for this case would be to measure KLD along the geodesic connecting the two models. Here, however, we will not concern ourselves with this problem.

5.3 Fisher Information Matrix

Consider the problem of estimating a parameter θ of a probability density function (pdf) $f(\boldsymbol{t}|\theta)$ using the available data. An estimator of θ is a function $\hat{\theta}(\boldsymbol{t})$ of the available data $\boldsymbol{t}$. To assess the error of the estimator, we calculate the mean squared error $MSE(\hat{\theta}) = E[\hat{\theta}(\boldsymbol{t}) - \theta]^2$, E denoting expectation. It is a well known fact that the mean squared error can be decomposed into a sum of variance and squared bias of the estimator [Bishop, 1996]:

$$MSE(\hat{\theta}) = var(\hat{\theta}(\boldsymbol{t})) + (E[\hat{\theta}(\boldsymbol{t})] - \theta)^2. \tag{5.22}$$

In the special case where we are dealing with an unbiased estimator $\hat{\theta}(\boldsymbol{t})$, i.e. $E[\hat{\theta}(\boldsymbol{t})] = \theta$, the bias term of the error becomes zero and the variance becomes equal to the mean squared error of the estimator. In [Frieden, 1998] an important result is proved that bounds error MSE. If $\hat{\theta}(\boldsymbol{t})$ is unbiased, we have:

$$E[\hat{\theta}(\boldsymbol{t})] - \theta = 0 \Leftrightarrow E[(\hat{\theta}(\boldsymbol{t}) - \theta)] = \int_{\mathcal{T}} (\hat{\theta}(\boldsymbol{t}) - \theta) f(\boldsymbol{t}|\theta) \boldsymbol{dt} = 0. \tag{5.23}$$

Differentiation of both sides follows [1]:

$$\int_{\mathcal{T}} (\hat{\theta}(\boldsymbol{t}) - \theta) \frac{df(\boldsymbol{t}|\theta)\boldsymbol{dt}}{d\theta} - \int_{\mathcal{T}} f(\boldsymbol{t}|\theta)\boldsymbol{dt} = 0. \tag{5.24}$$

Since f is a pdf, we have:

$$\int_{\mathcal{T}} (\hat{\theta}(\boldsymbol{t}) - \theta) \frac{df(\boldsymbol{t}|\theta)}{d\theta}\boldsymbol{dt} = 1. \tag{5.25}$$

Introduce the logarithm:

$$\int_{\mathcal{T}} (\hat{\theta}(\boldsymbol{t}) - \theta) f(\boldsymbol{t}|\theta) \frac{d\log f(\boldsymbol{t}|\theta)}{d\theta}\boldsymbol{dt} = 1. \tag{5.26}$$

Finally, we factorise and use the Schwartz-Cauchy inequality:

$$\begin{gathered} \int_{\mathcal{T}} (\hat{\theta}(\boldsymbol{t}) - \theta)\sqrt{f(\boldsymbol{t}|\theta)} \frac{d\log f(\boldsymbol{t}|\theta)}{d\theta}\sqrt{f(\boldsymbol{t}|\theta)}\boldsymbol{dt} = 1, \\ \left[\int_{\mathcal{T}} (\hat{\theta}(\boldsymbol{t}) - \theta)^2 f(\boldsymbol{t}|\theta)\boldsymbol{dt}\right] \left[\int_{\mathcal{T}} (\frac{d\log f(\boldsymbol{t}|\theta)}{d\theta})^2 f(\boldsymbol{t}|\theta)\boldsymbol{dt}\right] \geq 1 \end{gathered} \tag{5.27}$$

We now discern the following two terms in the brackets on the left-hand-side:

$$MSE(\hat{\theta})F(\theta) \geq 1. \tag{5.28}$$

This important result, known as the Cramer-Rao inequality, holds for the mean squared error of unbiased estimators. In (5.28) F is the *Fisher information* and it is defined as:

$$F(\theta) = -E_{\boldsymbol{t}}[(\frac{d}{d\theta} log f(\boldsymbol{t}|\theta))^2]. \tag{5.29}$$

Furthermore in [Degroot, 1996], an alternative form of the Fisher information is given that is based on the second order derivatives of the log-likelihood:

$$F(\theta) = -E_{\boldsymbol{t}}[\frac{d^2}{d\theta^2}\log f(\boldsymbol{t}|\theta)]. \tag{5.30}$$

Regarding the multivariate case, when $\boldsymbol{\theta} = \{\theta_1, ..., \theta_N\}$ the Fisher information matrix is defined in [Myung and Daniel, 2005] as the $N \times N$ matrix with entries $i, j = 1, \ldots, N$:

$$\boldsymbol{F}(\boldsymbol{\theta})_{i,j} = E_{\boldsymbol{t}}[\frac{\partial^2}{\partial\theta_i \partial\theta_j} log f(\boldsymbol{t}|\boldsymbol{\theta})], \tag{5.31}$$

In [Frieden, 1998] quantity F is interpreted as the *quality of a measuring process*. Thus, since

[1] assuming that the necessary conditions for differentiating inside the integral are met.

F (or $\det(\mathbf{F})$ in the multivariate case) is reciprocal to the error in (5.28), the error of the measurement increases as F decreases and vice versa. An example is presented in [Degroot, 1996] where the Fisher information of a one-dimensional Gaussian distribution $\mathcal{N}(t; \mu, \sigma^2)$ is considered. The distribution is of known variance σ^2 and we want to determine the mean μ. The log-likelihood of the distribution is:

$$\log \mathcal{L}(\mu) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{(x-\mu)^2}{2\sigma^2},$$

and its second derivative is:

$$\frac{\partial^2}{\partial \mu^2}\log \mathcal{L}(\mu) = -\frac{1}{\sigma^2}.$$

Thus, the Fisher information for parameter μ is:

$$F(\mu) = \frac{1}{\sigma^2}.$$

This means that the error of an unbiased estimator of μ is bound by the variance of the distribution, $MSE(\hat{\theta}) \geq \sigma^2$. This result makes indeed intuitive sense; if we try to estimate the mean by sampling, naturally we expect that for a "wide" Gaussian our estimate will be susceptible to high error. Conversely, for a "narrow" Gaussian we expect our estimate to be more accurate.

It is worth noting that maximum-likelihood estimators are asymptotically unbiased and normally distributed. In particular the variance of the distribution of a maximum likelihood estimator is equal to the Fisher information matrix [Kay, 1993]:

$$(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}) \sim \mathcal{N}(0, \boldsymbol{F}(\boldsymbol{\theta})). \tag{5.32}$$

Thus, the Fisher information can also be regarded as a quality measure for maximum likelihood estimation.

As we will see in the following sections, Fisher information is linked to the geometry of the model space $\mathcal{M}$. Thus, it constitutes a tool useful in considering magnification factors on $\mathcal{M}$.

5.4 Definition of a Manifold

Although we have not explicitly introduced the notion of a manifold so far, we have made use of it numerous times in chapter 4. A familiar example of a manifold is the boundary of a solid object. Though encountered in our three-dimensional space, when inspected closely it looks like a flat plane. Another example is that of the locally flat appearance of the spherical

earth. Mathematically speaking, the surface of the earth is a subset of $\mathbb{R}^3$ that locally resembles $\mathbb{R}^2$. More precisely, this local resemblance means that the neighbourhood $U_{\boldsymbol{a}}$ of every point $\boldsymbol{a}$ on the surface, must be *homeomorphic* to an open set in $\mathbb{R}^2$. For two sets U and V to be homeomorphic, a continuous and bijective function $U \to V$ must exist, whose inverse $V \to U$ is also continuous. Such a homeomorphic function $\tau : U_{\boldsymbol{a}} \to \mathbb{R}^2$ is called a chart. A chart provides a coordinate system that applies locally at point $\boldsymbol{a}$. These definitions can be generalised from the two dimensional surface to a d-dimensional object called a *manifold* [Small, 1996]. We denote a manifold by $\mathcal{M}$. On a d-dimensional manifold $\mathcal{M}$, each neighbourhood U is homeomorphic to an open set in Euclidean space $\mathbb{R}^d$.

However a manifold is more than merely a collection of open subsets, local patches, that resemble $\mathbb{R}^d$. An additional criterion [Small, 1996] is necessary that states how these local patches combine to create the manifold. Specifically, it is the overlapping of neighbouring patches that needs to be addressed. The overlapping area $U \cup V$ is addressed by both coordinate systems τ and ξ. We require the coordinate systems to be consistent with each other. Technically, it is required that:

$$\xi \circ \tau^{-1} : \tau(U \cup V) \to \xi(U \cup V), \tag{5.33}$$

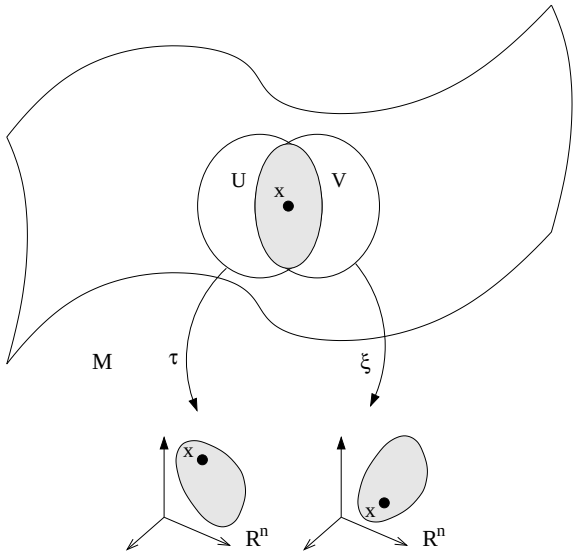


Fig. 5.4: Two overlapping local patches U and V on manifold $\mathcal{M}$. Patches U and V are homeomorphic to a Euclidean space $\mathbb{R}^n$ via charts τ and ξ respectively. Moreover, U and V are compatible charts via the homeomorphism $\xi \circ \tau^{-1}$.

be a homeomorphism [Small, 1996]. Such a requirement ensures that patches are "glued" together so as to form manifold $\mathcal{M}$. The situation is illustrated in Fig. 5.4. However, this re-

quirement may be extended to further ensure differentiability on $\mathcal{M}$. To ensure differentiability, functions in (5.33) are required to be *diffeomorphic* [Small, 1996]. A function is diffeomorphic when it is bijective, differentiable and has a differentiable inverse. The property of diffeomorphism establishes manifold $\mathcal{M}$ as a differentiable manifold.

5.5 Manifold of Statistical Models

In [Amari, 1959] the structure of an n-dimensional manifold is introduced in probability distributions. We consider a space $\mathcal{M}$ where probabilistic models of the form $p(\boldsymbol{t}|\boldsymbol{\theta})$ reside: $\mathcal{M} = \{p(\boldsymbol{t}|\boldsymbol{\theta})\}$ where $\boldsymbol{t}$ is a random variable on a sample space $\mathcal{T}$ and $\boldsymbol{\theta}$ is the n-dimensional parameter vector of the model. A mapping $\xi : \mathcal{M} \rightarrow \mathbb{R}^n$, i.e. $\xi(p(\boldsymbol{t}|\boldsymbol{\theta})) = \boldsymbol{\theta}$, is defined that plays the role of a coordinate chart. Thus, for each model $p(\boldsymbol{t}|\boldsymbol{\theta})$ its parameter vector $\boldsymbol{\theta}$ plays the role of a coordinate vector. This introduces a differentiable structure which makes $\mathcal{M}$ a differentiable manifold. As an example we consider a statistical manifold $\mathcal{M}$ of Gaussian distributions [Amari, 1959]. $\mathcal{M}$ is composed of all Gaussian distributions $\mathcal{N}(\boldsymbol{t}; \mu, \sigma)$ and each Gaussian distribution is addressed on $\mathcal{M}$ by coordinate vector $\boldsymbol{\theta} = (\mu, \sigma)$.

As aforementioned in section 5.2, the KLD between two distributions $P(\cdot)$ and $Q(\cdot)$ is defined as:

$$D_{KL}\left[P||Q\right] = \int_{\mathcal{T}} P(\boldsymbol{t}) \log \frac{P(\boldsymbol{t})}{Q(\boldsymbol{t})} \boldsymbol{dt}.$$

We now consider the KLD between a distribution $p(\cdot|\boldsymbol{\theta})$, and its infinitesimal perturbation $p(\cdot|\boldsymbol{\theta} + \boldsymbol{d\theta})$ on $\mathcal{M}$:

$$D_{KL}\left[p(\cdot|\boldsymbol{\theta})||p(\cdot|\boldsymbol{\theta} + \boldsymbol{d\theta})\right] = \int_{\mathcal{T}} p(\boldsymbol{t}|\boldsymbol{\theta}) \log \frac{p(\boldsymbol{t}|\boldsymbol{\theta})}{p(\boldsymbol{t}|\boldsymbol{\theta} + \boldsymbol{d\theta})} \boldsymbol{dt}. \tag{5.34}$$

Since we are interested in the KLD between $p(\cdot|\boldsymbol{\theta})$ and perturbed versions of it, it is convenient to define the following divergence function:

$$g(\boldsymbol{d\theta}) \equiv D_{KL}\left[p(\cdot|\boldsymbol{\theta})||p(\cdot|\boldsymbol{\theta} + \boldsymbol{d\theta})\right]. \tag{5.35}$$

Following [Calmet and Calmet, 2005], we take a second-order Taylor expansion of the new distance function in the neighbourhood of $\boldsymbol{d\theta}$:

$$g(\boldsymbol{d\theta}) = g(\mathbf{0}) + \nabla_{\mathbf{0}} g(\boldsymbol{d\theta})(\boldsymbol{d\theta}) + \frac{1}{2} \boldsymbol{d\theta}^T \nabla^2_{\mathbf{0}} g(\boldsymbol{d\theta}) \boldsymbol{d\theta}. \tag{5.36}$$

Since $D_{KL}\left[p(\cdot|\boldsymbol{\theta})||p(\cdot|\boldsymbol{\theta} + \mathbf{0})\right] = 0$, it follows that $g(\mathbf{0}) = 0$. However, since $g(\mathbf{0}) = 0$ which is

the minimum value of the function, it follows that $g(\mathbf{0})$ is an inflexion point, thus $\nabla_{\mathbf{0}} g(\boldsymbol{d\theta}) = 0$. The second order gradient of $g(\boldsymbol{d\theta})$ is:

$$\nabla^2 g(\boldsymbol{d\theta}) \quad = \quad -\nabla \int_{\mathcal{T}} p(\boldsymbol{t}|\boldsymbol{\theta}) \nabla \log p(\boldsymbol{t}|\boldsymbol{\theta} + \boldsymbol{d\theta}) \boldsymbol{dt} = -E[\nabla^2 \log p(\boldsymbol{t}|\boldsymbol{\theta} + \boldsymbol{d\theta})], \tag{5.37}$$

which we identify as the negative Fisher information given by (5.31). Thus the KLD between a distribution and its infinitesimally perturbed version is measured as:

$$D_{KL}[p(\cdot|\boldsymbol{\theta})||p(\cdot|\boldsymbol{\theta} + \boldsymbol{d\theta})] = \frac{1}{2} \boldsymbol{d\theta}^T \boldsymbol{F}(\boldsymbol{\theta}) \boldsymbol{d\theta}. \tag{5.38}$$

Thus, the Fisher information matrix (FIM) acts as a metric tensor on the manifold of probability distributions parametrised by $p(\cdot|\boldsymbol{\theta})$.

5.6 Fisher Information as a Magnification Factor for GTM and Extensions

In the following sections we shall calculate the FIM, that acts as a metric tensor, for the original GTM and its extensions discussed in chapter 4. The FIM is calculated locally at each latent point $\boldsymbol{x}$. Determination of FIM allows us to calculate distances between model $p(\cdot|\boldsymbol{x})$ addressed by $\boldsymbol{x}$ and model $p(\cdot|\boldsymbol{dx})$ addressed by $\boldsymbol{x} + \boldsymbol{dx}$. Such distances are calculated by:

$$D_{KL}[p(\cdot|\boldsymbol{x})||p(\cdot|\boldsymbol{x} + \boldsymbol{dx}))] = \frac{1}{2} \boldsymbol{dx}^T \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{dx}. \tag{5.39}$$

The situation is illustrated in Fig. 5.5. The same perturbation scheme as in the KLD approximation method for estimating magnification factors, is employed here. We note, however, that unlike the KLD approximation this method constitutes an analytical way that explicitly takes into consideration the manifold on which the local noise models lie and provides a more theoretically satisfying approach.

5.6.1 FIM for Original GTM

We re-derive the magnification factors for the original GTM in [Bishop et al., 1997] using the concept of FIM. In GTM, as discussed in section 4.1, the local noise models are spherical Gaussians of common variance σ^2. The log-likelihood for a vector input $\boldsymbol{t}$ under model $p(\cdot|\boldsymbol{x})$ is:

$$\log p(\cdot|\boldsymbol{x}) \quad = \quad \log \frac{1}{\sigma\sqrt{2\pi}} - \frac{1}{2\sigma^2} (\boldsymbol{t} - \boldsymbol{\mu_x})^T (\boldsymbol{t} - \boldsymbol{\mu_x}). \tag{5.40}$$

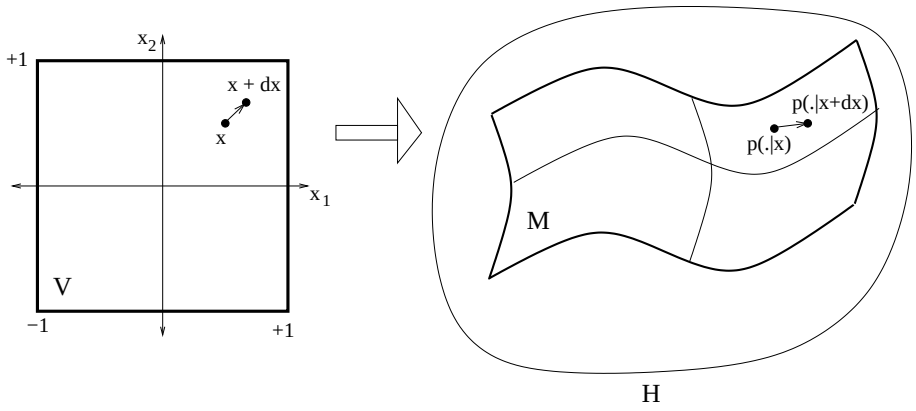


Fig. 5.5: Two-dimensional manifold $\mathcal{M}$ of local noise models $p(\cdot|\boldsymbol{x})$ parametrised by the latent space $\mathcal{V}$ (through (4.17)-(4.19) in the case of GTM-HMTM). The manifold is embedded in manifold $\mathcal{H}$ of all noise models of the same form. Latent coordinates $\boldsymbol{x}$ are displaced to $\boldsymbol{x}+\boldsymbol{dx}$. Kullback-Leibler divergence $D_{KL}[p(\cdot|\boldsymbol{x})||p(\cdot|\boldsymbol{x}+\boldsymbol{dx})]$ between the corresponding noise models $p(\cdot|\boldsymbol{x}), (p(\cdot|\boldsymbol{x}+\boldsymbol{dx}) \in \mathcal{M}$ can be determined via Fisher information matrix $\boldsymbol{F(x)}$ that acts like a metric tensor on the Riemannian manifold $\mathcal{M}$.

The second order derivatives of the log-likelihood of model $p(\cdot|\boldsymbol{x})$ with respect to the parameters x_r, x_s with $r, s \in 1, 2$ are:

$$\begin{aligned}
\frac{\partial^2}{\partial x_r \partial x_s} \log p(\cdot|\boldsymbol{x}) &= \frac{\partial^2}{\partial x_r \partial x_s}\left(\log \frac{1}{\sigma\sqrt{2\pi}} - \frac{1}{2\sigma^2}(\boldsymbol{t}-\boldsymbol{\mu_x})^T(\boldsymbol{t}-\boldsymbol{\mu_x})\right), \\
&= \frac{\partial}{\partial x_s}\left(\frac{1}{\sigma^2}(\boldsymbol{t}-\boldsymbol{\mu_x})^T \boldsymbol{W} \frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})\right), \\
&= -\frac{1}{\sigma^2}\frac{\partial}{\partial x_s}\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{W}^T \boldsymbol{W} \frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) + \frac{1}{\sigma^2}(\boldsymbol{t}-\boldsymbol{\mu_x})^T \boldsymbol{W} \frac{\partial^2}{\partial x_r \partial x_s}\boldsymbol{\phi}(\boldsymbol{x}).
\end{aligned} \tag{5.41}$$

Using the definition of FIM in (5.31), the elements of FIM are:

$$\begin{aligned}
\boldsymbol{F}(\boldsymbol{x})_{r,s} &= -\int_{\mathcal{T}} \frac{\partial^2}{\partial x_r \partial x_s} \log p(\boldsymbol{t}|\boldsymbol{x}) \boldsymbol{dt} \\
&= -\int_{\mathcal{T}} -\frac{1}{\sigma^2}\frac{\partial}{\partial x_s}\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{W}^T \boldsymbol{W} \frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) + \frac{1}{\sigma^2}(\boldsymbol{t}-\boldsymbol{\mu_x}) \boldsymbol{W} \frac{\partial^2}{\partial x_r \partial x_s}\boldsymbol{\phi}(\boldsymbol{x}) \boldsymbol{dt} \\
&= \frac{1}{\sigma^2}\frac{\partial}{\partial x_s}\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{W}^T \boldsymbol{W} \frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) + \int_{\mathcal{T}} \frac{1}{\sigma^2}(\boldsymbol{t}-\boldsymbol{\mu_x})^T \boldsymbol{W} \frac{\partial^2}{\partial x_r \partial x_s}\boldsymbol{\phi}(\boldsymbol{x}) \boldsymbol{dt} \\
&= \frac{1}{\sigma^2}\frac{\partial}{\partial x_s}\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{W}^T \boldsymbol{W} \frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}).
\end{aligned} \tag{5.42}$$

This formula is in correspondence (discarding constants) to the magnification factors obtained via the original derivation in section 5.1 (where the ratio of areas was considered) and the KLD approximation in section 5.2.1.

5.6.2 FIM for GTM-HMM

In [Tiňo et al., 2004] GTM is extended to the visualisation of symbolic sequences using HMMs as noise models. Each point $\boldsymbol{x} \in \mathcal{V}$ is mapped to a set of HMM parameters that address an HMM in the constrained two-dimensional manifold $\mathcal{M}$. The family of HMMs considered are HMMs that emit discrete symbols from an alphabet $\mathcal{A}$ of A number of discrete symbols $\boldsymbol{s}_t$. The respective mappings for generating the initial, transition, and emission parameters are:

- Initial probabilities:

$$\boldsymbol{\pi}(\boldsymbol{x}) = \{p(q_1 = k|\boldsymbol{x})\}_{k=1,\ldots,K} = \{g_k(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\}_{k=1,\ldots,K}, \tag{5.43}$$

- Transition probabilities:

$$\boldsymbol{B}(\boldsymbol{x}) = \{p(q_t = l|q_{t-1} = k, \boldsymbol{x})\}_{k=1,\ldots,K,l=1,\ldots,K} = \{g_l(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\}_{k=1,\ldots,K}, \tag{5.44}$$

- Emission probabilities:

$$\boldsymbol{\Psi}(\boldsymbol{x}) = \{p(\boldsymbol{s}_t = s|q_{t-1} = k, \boldsymbol{x})\}_{s=1,\ldots,S,k=1,\ldots,K} = \{g_l(\boldsymbol{W}^{(\boldsymbol{\psi}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\}_{s=1,\ldots,S,k=1,\ldots,K.} \tag{5.45}$$

Thus, each point $\boldsymbol{x} \in \mathcal{V}$ is mapped to a set of HMM parameters that address an HMM in space $\mathcal{M}$. We would like to calculate the local FIM and use it as the metric tensor to estimate the local magnification factors. Unfortunately, there is no closed-form formula for calculating FIM for HMMs. However in [Lystig and Hughes, 2002], a framework for the efficient calculation of the *observed* FIM of HMMs is presented. It is based on a variant of the forward algorithm (see (3.45)-(3.47)) that is immune to numerical underflow. We adapt the framework to the special kind of GTM-parametrisation of HMMs expressed by (5.43)-(5.45).

We first calculate the likelihood using this revised forward algorithm. Similar to the forward algorithm, the revised version is a recursive process that starts from the beginning of the sequence. Notation $\lambda_k(t; \boldsymbol{x})$ that follows below, denotes the partially calculated likelihood of the HMM addressed by latent point $\boldsymbol{x}$, where parameter $t = 1, 2, \ldots, T$ indexes the time (position) that a symbol occurs, while subindex k denotes the current state $k = 1, 2, \ldots, K$.

- Initial step, at first time step t=1:

$$\lambda_k(1; \boldsymbol{x}) = p(\boldsymbol{s}_1|q_1 = k, \boldsymbol{x})p(q_1 = k|\boldsymbol{x}). \tag{5.46}$$

- Recursive step, at time steps $t = 2, \ldots, T$:

$$\begin{aligned} \lambda_k(t; \boldsymbol{x}) &= p(\boldsymbol{s}_t, q_t = k | \boldsymbol{s}_1, \ldots, \boldsymbol{s}_{t-1}, \boldsymbol{x}) \\ &= \sum_{i=1}^{K} [\lambda_i(t-1; \boldsymbol{x}) p(\boldsymbol{s}_t | q_t = k, \boldsymbol{x}) p(q_t = k | q_{t-1} = i, \boldsymbol{x})] (\Lambda(t-1; \boldsymbol{x}))^{-1}, \end{aligned} \tag{5.47}$$

where $\Lambda(t; \boldsymbol{x}) = \sum_{j=1}^{K} \lambda_j(t; \boldsymbol{x})$. Thus, the model log-likelihood can be written as:

$$\log \mathcal{L}(\boldsymbol{x} | \mathcal{D}) = \sum_{t=1}^{T} \log \Lambda(t; \boldsymbol{x}). \tag{5.48}$$

Of course, this quantity is in theory identical to the result of the forward algorithm, but in practice it may be more accurate due to numerical instabilities.

What are the sequences used in estimating the local Fisher information matrix at point $\boldsymbol{x}$? As we saw in (5.31), FIM is defined as the expectation of all observations generated by the considered probabilistic model $p(\cdot|\boldsymbol{x})$ at hand. Hence, when estimating the local FIM at latent point $\boldsymbol{x}$, we generate a set of sequences from the HMM $p(\cdot|\boldsymbol{x})$ addressed by $\boldsymbol{x}$. We denote this set of sequences by $\mathcal{S}(\boldsymbol{x})$. We assume that the sequences in $\mathcal{S}(\boldsymbol{x}) = \{\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(N)}\}$ are of length T.

Moreover, the FIM in (5.31) requires the 2-nd order derivatives of the log-likelihood which is calculated by (5.48). Based on this revised forward algorithm, we need the 2-nd order derivatives of (5.48) with respect to the coordinates of latent point $\boldsymbol{x}$. However, before we calculate the 2-nd order derivatives of (5.48), it is necessary to calculate the 1-st order derivatives of (5.48). These derivatives, as we shall see, require in turn the calculation of further derivatives, namely the 1-st and 2-nd order derivatives of the parameters of $p(\cdot|\boldsymbol{x})$ with respect to the coordinates of $\boldsymbol{x}$.

We commence the calculation of the aforementioned menagerie of derivatives with the 1-st order derivatives.

1-st Order Derivatives

In this step, as in the forward algorithm, we start at the beginning of the sequence, recursively evaluating the 1-st order derivatives of the revised likelihood with respect to the coordinates of latent point $\boldsymbol{x}$. In this setting where the latent space $\mathcal{V}$ is a two-dimensional space, each latent point has two coordinates $x_r, r = 1, 2$:

- Initial step, at first time step t=1:

$$\begin{aligned}\xi_k(1;r,\boldsymbol{x}) &= \frac{\partial}{\partial x_r}p(\boldsymbol{s}_1|q_1=k,\boldsymbol{x})p(q_1=k|\boldsymbol{x}) \\ &= [\frac{\partial}{\partial x_r}p(\boldsymbol{s}_1|q_1=k,\boldsymbol{x})]p(q_1=k|\boldsymbol{x}) + p(\boldsymbol{s}_1|q_1=k,\boldsymbol{x})[\frac{\partial}{\partial x_r}p(q_1=k|\boldsymbol{x})].\end{aligned} \tag{5.49}$$

- Recursive step, at time steps $t=2,\ldots,T$:

$$\begin{aligned}\xi_k(t;r,\boldsymbol{x}) &= \sum_{i=1}^{T}\Big\{\xi_i(t-1;x_r,\boldsymbol{x})p(\boldsymbol{s}_t|q_t=k,\boldsymbol{x})p(q_t=i|q_{t-1}=k,\boldsymbol{x}) \\ &+ \lambda_i(t-1;\boldsymbol{x})[\frac{\partial}{\partial x_r}p(\boldsymbol{s}_t|q_t=k,\boldsymbol{x})]p(q_t=k|q_{t-1}=i,\boldsymbol{x}) \\ &+ \lambda_i(t-1;\boldsymbol{x})p(\boldsymbol{s}_t|q_t=k,\boldsymbol{x})[\frac{\partial}{\partial x_r}p(q_t=k|q_{t-1}=i,\boldsymbol{x})]\Big\}(\Lambda(t-1;\boldsymbol{x}))^{-1}.\end{aligned} \tag{5.50}$$

The 1-st order derivative of the log-likelihood with respect to x_r is then:

$$\frac{\partial}{\partial x_r}\log\mathcal{L}(\boldsymbol{x}|\mathcal{D}) = \frac{\Xi(T;r,\boldsymbol{x})}{\Lambda(T;\boldsymbol{x})}, \tag{5.51}$$

where $\Xi(t;r,\boldsymbol{x}) = \sum_{j=1}^{K}\xi_j(t;r,\boldsymbol{x})$.

The calculations recursively employ $\xi_i(t;r,\boldsymbol{x})$, $\lambda_i(t;\boldsymbol{x})$ and $\Lambda(t;\boldsymbol{x})$. They also employ the 1-st order derivatives of the HMM parameters of $p(\cdot|\boldsymbol{x})$ induced by the GTM-HMM, namely the derivatives of the initial probabilities $\frac{\partial}{\partial x_r}p(\boldsymbol{s}_1|q_1=k,\boldsymbol{x})$, transition probabilities $\frac{\partial}{\partial x_r}p(q_t=k|q_{t-1}=i,\boldsymbol{x})$ and emission probabilities $\frac{\partial}{\partial x_r}p(\boldsymbol{s}_t|q_t=k,\boldsymbol{x})$. We proceed with the calculation of the derivatives of the HMM parameters of $p(\cdot|\boldsymbol{x})$.

- 1-st order derivative of initial probability for state k under the parametrisation in (5.43):

$$\begin{aligned}\frac{\partial}{\partial x_r}p(h_1=k|\boldsymbol{x}) &= \frac{\partial}{\partial x_r}g_k(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x})) = \frac{\partial}{\partial x_r}\frac{\exp(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))}{\sum_{i=1}^{K}\exp(\boldsymbol{W}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))} \\ &= g_k(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[g_i(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]).\end{aligned} \tag{5.52}$$

- 1-st order derivative of transition from state k to state l under the parametrisation in

(5.44):

$$\begin{aligned}\frac{\partial}{\partial x_r}p(q_t = l|q_{t-1} = k, \boldsymbol{x}) &= g_l(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x})) \\ &\times \left(\boldsymbol{W}_l^{(\boldsymbol{B}_k)}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[g_i(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{B}_k)}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\right).\end{aligned} \tag{5.53}$$

- 1-st order derivative of emission probability of symbol s at state k under the parametrisation in (5.45):

$$\begin{aligned}\frac{\partial}{\partial x_r}p(\boldsymbol{s}_t^{(n)} = s|q_t = k, \boldsymbol{x}) &= g_s(\boldsymbol{W}^{(\boldsymbol{\psi}_k)}\boldsymbol{\phi}(\boldsymbol{x})) \\ &\times \left(\boldsymbol{W}_s^{(\boldsymbol{\psi}_k)}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{q}[g_i(\boldsymbol{W}^{(\boldsymbol{\psi}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{\psi}_k)}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\right).\end{aligned} \tag{5.54}$$

Moreover, in the preceding equations, the calculation of the 1-order derivatives of the basis functions, $\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})$ is required. We have:

$$\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) = \left[-\frac{1}{\sigma^2}\phi_m(\boldsymbol{x})(x_r - \mu_{m,r})\right]. \tag{5.55}$$

More detailed derivations are presented in Appendix D. We proceed with the calculation of 2-nd order derivatives.

2-nd Order Derivatives

We calculate the 2-nd order derivatives of the revised forward algorithm again in a recursive fashion, with respect to the r-th and h-th coordinates of latent point $\boldsymbol{x}$. We start at the beginning of the sequence:

- Initial step, at time step $t = 1$:

$$\begin{aligned}\omega_k(1; h, r, \boldsymbol{x}) &= [\frac{\partial^2}{\partial \boldsymbol{x}_h \partial \boldsymbol{x}_r}p(\boldsymbol{s}_1|q_1 = k, \boldsymbol{\theta})]p(q_1 = k|\boldsymbol{\theta}) + [\frac{\partial}{\partial \boldsymbol{x}_h}p(\boldsymbol{s}_1|q_1 = k, \boldsymbol{\theta})][\frac{\partial}{\partial \boldsymbol{x}_r}p(q_1 = k|\boldsymbol{\theta})] \\ &+ [\frac{\partial}{\partial \boldsymbol{x}_r}p(\boldsymbol{s}_1|q_1 = k, \boldsymbol{\theta})][\frac{\partial}{\partial \boldsymbol{x}_h}[p(q_1 = k|\boldsymbol{\theta})] + p(\boldsymbol{s}_1|q_1 = k, \boldsymbol{\theta})[\frac{\partial^2}{\partial \boldsymbol{x}_h \partial \boldsymbol{x}_r}p(q_1 = k|\boldsymbol{\theta})].\end{aligned} \tag{5.56}$$

- Recursive step, at time steps $t = 2, \ldots, T$:

$$\begin{aligned}
\omega_k(t; h, r, \boldsymbol{x}) &= \sum_{i=1}^{K} \omega_i(t-1; h, r, \boldsymbol{x}) p(\boldsymbol{s}_t | q_t = k, \boldsymbol{x}) p(q_t = k | q_{t-1} = i, \boldsymbol{x}) \\
&+ \psi_i(t-1, h, \boldsymbol{x}) [\frac{\partial}{\partial x_r} p(\boldsymbol{s}_t | q_t = k, \boldsymbol{x})] p(q_t = k | q_{t-1} = i, \boldsymbol{x}) \\
&+ \psi_i(t-1, r, \boldsymbol{x}) [\frac{\partial}{\partial x_h} p(\boldsymbol{s}_t | q_t = k, \boldsymbol{x})] p(q_t = k | q_{t-1} = i, \boldsymbol{x}) \\
&+ \lambda_i(t-1; \boldsymbol{x}) [\frac{\partial}{\partial x_h} p(\boldsymbol{s}_t | q_t = k, \boldsymbol{x})] [\frac{\partial}{\partial x_r} p(q_t = k | q_{t-1} = i, \boldsymbol{x}) \\
&+ \lambda_i(t-1; \boldsymbol{x}) [\frac{\partial}{\partial x_r} p(\boldsymbol{s}_t | q_t = k, \boldsymbol{x})] [\frac{\partial}{\partial x_h} p(q_t = k | q_{t-1} = i, \boldsymbol{x}) \\
&+ \lambda_i(t-1; \boldsymbol{x}) p(\boldsymbol{s}_t | q_t = k, \boldsymbol{x})] [\frac{\partial^2}{\partial x_h \partial x_r} p(q_t = k | q_{t-1} = i, \boldsymbol{x}) (\Lambda(t-1; \boldsymbol{x}))^{-1}.
\end{aligned} \tag{5.57}$$

The 2-nd order derivative of the log-likelihood with respect to x_r, x_h is then:

$$\frac{\partial^2}{\partial x_h \partial x_r} \log \mathcal{L}(\boldsymbol{x} | \mathcal{D}) = \mathcal{Q}_{h,r}(\boldsymbol{x}) = \frac{\Omega(T; h, r, \boldsymbol{x})}{\Lambda(T; \boldsymbol{x})} - \frac{\Xi(T; h, \boldsymbol{x}) \Xi(T; r, \boldsymbol{x})}{(\Lambda(T; \boldsymbol{x}))^2}, \tag{5.58}$$

where $\Omega(t; h, r, \boldsymbol{x}) = \sum_{t=1}^{K} \omega_t(t; h, r, \boldsymbol{x})$.

After these calculations, we obtain the elements of the observed Fisher information matrix $\hat{\boldsymbol{F}}(\boldsymbol{x})$, given the set of sequences $\mathcal{S}(\boldsymbol{x})$, as:

$$\hat{\boldsymbol{F}}(\boldsymbol{x})_{h,r} = -\frac{1}{N} \sum_{n=1}^{N} \mathcal{Q}_{h,r}^{(n)}(\boldsymbol{x}), \tag{5.59}$$

where we have augmented the notation of quantity $\mathcal{Q}_{h,r}(\boldsymbol{x})$ with index n to denote the calculation of the 2-nd order derivative of the log-likelihood for the n-th sequence in set $\mathcal{S}(\boldsymbol{x}) = \{\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(N)}\}$.

The preceding calculations recursively employ $\omega(t-1; h, r, \boldsymbol{x})$, $\psi(t-1; h, \boldsymbol{x})$, $\lambda(t-1; \boldsymbol{x})$ and $\Lambda(t-1; \boldsymbol{\theta})$. Moreover, they employ 2-nd order derivatives of the HMM parameters induced by GTM-HMM, namely derivatives of the initial probabilities $\frac{\partial^2}{\partial x_h \partial x_r} p(h_1 = k | \boldsymbol{x})$, the transition probabilities $\frac{\partial^2}{\partial x_h \partial x_r} p(q_t = l | q_{t-1} = k, \boldsymbol{x})$ and the emission probabilities $\frac{\partial^2}{\partial x_h \partial x_r} p(\boldsymbol{s}_t^{(n)} = s | q_t = k, \boldsymbol{x})$. We proceed with the calculation the 2-nd order derivatives of the induced HMM parameters.

- 2-nd order derivative of initial probability for state k under the parametrisation in (5.43):

$$\begin{aligned}\frac{\partial^2}{\partial x_h \partial x_r} p(h_1 = k|\boldsymbol{x}) &= \frac{\partial^2}{\partial x_h \partial x_r} g_k(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x})) = \frac{\partial}{\partial x_h}\left(\frac{\partial}{\partial x_r} g_k(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\right) \\ &= [\frac{\partial}{\partial x_h} g_k(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))](\boldsymbol{W}_k^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[g_i(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]) \\ &+ g_k(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\Bigg(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\frac{\partial^2}{\partial x_h \partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[\frac{\partial}{\partial x_h} g_i(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) \\ &+ g_i(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{\pi})}\frac{\partial^2}{\partial x_h \partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\Bigg). \end{aligned} \tag{5.60}$$

- 2-nd order derivative of transition probability from state k to state l under the parametrisation in (5.44):

$$\begin{aligned}&\frac{\partial^2}{\partial x_h \partial x_r} p(q_t = l|q_{t-1} = k, \boldsymbol{x}) = \\ &[\frac{\partial}{\partial x_h} g_l(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}))]\Bigg(\boldsymbol{W}_l^{(\boldsymbol{B}_k)}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[g_i(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{B}_k)}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\Bigg) \\ &+ g_l(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\Bigg(\boldsymbol{W}_l^{(\boldsymbol{B}_k)}\frac{\partial^2}{\partial x_h \partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[\frac{\partial}{\partial x_h} g_i(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{B}_k)}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) \\ &+ g_i(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{B}_k)}\frac{\partial^2}{\partial x_h \partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\Bigg). \end{aligned} \tag{5.61}$$

- 2-nd order derivative of emission probability for s at state k under the parametrisation in (5.45):

$$\begin{aligned}&\frac{\partial^2}{\partial x_h \partial x_r} p(\boldsymbol{s}_t^{(n)} = s|q_t = k, \boldsymbol{x}) = \\ &g_s(\boldsymbol{W}^{(\boldsymbol{\psi}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\Bigg(\boldsymbol{W}_s^{(\boldsymbol{\psi}_k)}\frac{\partial}{\partial x_h \partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[\frac{\partial}{\partial x_h} g_i(\boldsymbol{W}^{(\boldsymbol{\psi}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{\psi}_k)}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) \\ &+ g_i(\boldsymbol{W}^{(\boldsymbol{\psi}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{\psi}_k)}\frac{\partial^2}{\partial x_h \partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\Bigg). \end{aligned} \tag{5.62}$$

We conclude with the calculation of the 2-order derivatives of the basis functions, $\frac{\partial^2}{\partial x_h \partial x_r}\boldsymbol{\phi}(\boldsymbol{x})$, required in the preceding formulas. We have:

$$\frac{\partial^2}{\partial x_h \partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) = -\frac{1}{\sigma^2}\phi_m(\boldsymbol{x}) + (x_h - \mu_{m,h})(\boldsymbol{x})(x_r - \mu_{m,r})\frac{1}{\sigma^4}\phi_m. \tag{5.63}$$

More detailed derivations can be found in Appendix D.

5.6.3 FIM for GTM-HMTM

Magnification factors for GTM-HMTM are calculated in the same fashion as for GTM-HMM in the previous section 5.6.2. Of course in the GTM-HMTM setting, points $\boldsymbol{x}$ in latent space $\mathcal{V}$ are mapped to a $p(\cdot|\boldsymbol{x})$ which is an HMTM residing on a two-dimensional manifold $\mathcal{M}$. Again, in order to appreciate the magnification around a point $\boldsymbol{x}$, we need to estimate the local FIM. Just like in GTM-HMM, this requires the 1-st and 2-nd order derivatives of the model log-likelihood of $p(\cdot|\boldsymbol{x})$ with respect to the coordinates of $\boldsymbol{x}$. It also requires the 1-st and 2-nd order derivatives of the HMTM parameters with respect to the coordinates of $\boldsymbol{x}$. Moreover, as required by the Fisher information matrix in (5.31), the likelihood is evaluated over a set of sample trees $\mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ generated by $p(\cdot|\boldsymbol{x})$. First, the 1-st order derivatives are presented followed by the 2-nd order derivatives.

1-st Order Derivatives

Likelihood for HMTMs is calculated via the upward recursion [Crouse et al., 1998] that was presented in section 3.4.1. For ease of exposition, we restate here the steps of the upward recursion:

- The recursion starts from the leaves u of the tree:

$$\beta_k(u;\boldsymbol{x}) = p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x}). \tag{5.64}$$

- Recursive step for non-leaf nodes u:

$$\begin{aligned}
\beta_k(u;\boldsymbol{x}) &= p(\boldsymbol{y}_u|q_u = k, \boldsymbol{x}) = \Big\{ \prod_{v \in ch(u)} p(\boldsymbol{y}_v|q_u = k, \boldsymbol{x}) \Big\} p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x}) \\
&= \Big\{ \prod_{v \in ch(u)} \sum_{i=1}^{K} p(\boldsymbol{y}_v|q_v = i, \boldsymbol{x}) p(q_v = i|q_u = k, \boldsymbol{x}) \Big\} p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x}) \\
&= \Big\{ \prod_{v \in ch(u)} \sum_{i=1}^{K} \beta_i(v;\boldsymbol{x}) p(q_v = i|q_u = k, \boldsymbol{x}) \Big\} p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x}).
\end{aligned} \tag{5.65}$$

- Final step:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{k=1}^{K} \beta_k(1;\boldsymbol{x}) p(q_1 = k|\boldsymbol{x}). \tag{5.66}$$

Starting again from the leaves of tree $\boldsymbol{y}$, we recursively evaluate 1-st order derivatives of likelihood, based on the upward recursion, with respect to the latent coordinates x_1, x_2. Let $r \in \{1, 2\}$:

- The recursion starts from the leaves of the tree:

$$\frac{\partial}{\partial x_r}\beta_k(u;\boldsymbol{x}) = \frac{\partial}{\partial x_r}p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x}). \tag{5.67}$$

- Recursive step for non-leaf nodes u:

$$\begin{aligned}\frac{\partial}{\partial x_r}\beta_k(u,\boldsymbol{x}) &= \frac{\partial}{\partial x_r}p(\boldsymbol{y}_u|q_u = k, \boldsymbol{x}) = \left\{\frac{\partial}{\partial x_r}\gamma_k(u;\boldsymbol{x})\right\}p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x}) \\ &+ \gamma_k(u;\boldsymbol{x})\left\{\frac{\partial}{\partial x_r}p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x})\right\},\end{aligned} \tag{5.68}$$

where

$$\begin{aligned}\gamma_k(u;\boldsymbol{x}) &= \prod_{v\in ch(u)}\zeta_k(u,v;\boldsymbol{x}), \\ \zeta_k(u,v;\boldsymbol{x}) &= \sum_{i=1}^{K}\beta_i(v;\boldsymbol{x})p(q_v = i|q_u = k, \boldsymbol{x}), \\ \frac{\partial}{\partial x_r}\gamma_k(u;\boldsymbol{x}) &= \gamma_k(u;\boldsymbol{x})\sum_{v\in ch(u)}\frac{\partial}{\partial x_r}\log\zeta_k(u,v;\boldsymbol{x}), \\ \frac{\partial}{\partial x_r}\zeta_k(u,v;\boldsymbol{x}) &= \sum_{i=1}^{K}\frac{\partial}{\partial x_r}\beta_i(v;\boldsymbol{x})p(q_v = i|q_u = k, \boldsymbol{x}) + \beta_i(v;\boldsymbol{x})\frac{\partial}{\partial x_r}p(q_v = i|q_u = k, \boldsymbol{x}).\end{aligned} \tag{5.69}$$

- Final step:

$$\frac{\partial}{\partial x_r}p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{k=1}^{K}\left\{\frac{\partial}{\partial x_r}\beta_k(1;\boldsymbol{x})\right\}p(q_1 = k|\boldsymbol{x}) + \beta_k(1;\boldsymbol{x})\left\{\frac{\partial}{\partial x_r}p(q_1 = k|\boldsymbol{x})\right\}. \tag{5.70}$$

The above calculations depend on the 1-st order derivatives of initial state, state transition and state-conditional emission probabilities with respect to the latent coordinates:

- 1-st order derivative for initial probability for state k under the parametrisation in (4.18):

$$\frac{\partial}{\partial x_r} p(q_1 = k|\boldsymbol{x}) = g_k(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\Bigg(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\phi(\boldsymbol{x}) - \sum_{i=1}^{K} g_i(\boldsymbol{W}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\phi(\boldsymbol{x})\Bigg). \tag{5.71}$$

- 1-st order derivative for transition probability from state k to state l under the parametrisation in (4.19):

$$\begin{aligned}\frac{\partial}{\partial x_r} p(q_u = l|q_{\rho(u)} = k, \boldsymbol{x}) &= g_l(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\Bigg(\boldsymbol{W}_l^{(\boldsymbol{B}_k)}\frac{\partial}{\partial x_r}\phi(\boldsymbol{x}) \\ &- \sum_{i=1}^{K} g_i(\boldsymbol{W}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_i^{(\boldsymbol{B}_k)}\frac{\partial}{\partial x_r}\phi(\boldsymbol{x})\Bigg).\end{aligned} \tag{5.72}$$

- 1-st order derivative for means of the emission distribution for state k under the parametrisation in (4.17):

$$\frac{\partial}{\partial x_r}\boldsymbol{\mu}_k = \frac{\partial}{\partial x_r}\boldsymbol{W}^{(\boldsymbol{\psi}_k)}\phi(\boldsymbol{x}) = \boldsymbol{W}^{(\boldsymbol{\psi}_k)}\frac{\partial}{\partial x_r}\phi(\boldsymbol{x}). \tag{5.73}$$

The above derivatives regarding the initial state, state transition and emission parameters are calculated in the same fashion as the corresponding 1-st order derivatives in the case of GTM-HMM (Appendix D). The 1-st order derivatives of the basis function $\frac{\partial}{\partial x_r}\phi(\boldsymbol{x})$ are identical to (5.55).

2-nd Order Derivatives

We repeat the recursion once more, this time calculating the 2nd-order derivatives. Let $h, r \in \{1, 2\}$:

- The recursion starts from the leaves of the tree:

$$\frac{\partial^2}{\partial x_h \partial x_r}\beta_k(u; \boldsymbol{x}) = \frac{\partial^2}{\partial x_h \partial x_r} p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x}). \tag{5.74}$$

- Recursive step for non-leaf nodes u:

$$\begin{aligned}\frac{\partial^2}{\partial x_h \partial x_r}\beta_k(u;\boldsymbol{x}) &= \frac{\partial^2}{\partial x_h \partial x_r}p(\boldsymbol{y}_u|q_u=k,\boldsymbol{x}) \\ &= \frac{\partial^2}{\partial x_h \partial x_r}\gamma_k(u;\boldsymbol{x})p(\boldsymbol{o}_u|q_u=k,\boldsymbol{x}) + \frac{\partial}{\partial x_h}\gamma_k(u;\boldsymbol{x})\frac{\partial}{\partial x_r}p(\boldsymbol{o}_u|q_u=k,\boldsymbol{x}) \\ &+ \frac{\partial}{\partial x_r}\gamma_k(u;\boldsymbol{x})\frac{\partial}{\partial x_h}p(\boldsymbol{o}_u|q_u=k,\boldsymbol{x}) + \gamma_k(u;\boldsymbol{x})\frac{\partial^2}{\partial x_h \partial x_r}p(\boldsymbol{o}_u|q_u=k,\boldsymbol{x}),\end{aligned} \tag{5.75}$$

where

$$\begin{aligned}\frac{\partial^2}{\partial x_h \partial x_r}\gamma_k(u;\boldsymbol{x}) &= \frac{\partial}{\partial x_h}\left\{\gamma_k(u;\boldsymbol{x})\sum_{v\in ch(u)}\frac{\partial}{\partial x_r}\log\zeta_k(u,v;\boldsymbol{x})\right\} \\ &= \gamma_k(u;\boldsymbol{x})\left(\sum_{v\in ch(u)}\frac{\partial}{\partial x_h}\log\zeta_k(u,v;\boldsymbol{x})\right)\left(\sum_{v\in ch(u)}\frac{\partial}{\partial x_r}\log\zeta_k(u,v;\boldsymbol{x})\right) \\ &+ \gamma_k(u;\boldsymbol{x})\left(\sum_{i=1}^{K}\frac{-1}{(\zeta_k(u,v;\boldsymbol{x}))^2}\frac{\partial}{\partial x_h}\zeta_k(u,v;\boldsymbol{x})\frac{\partial}{\partial x_r}\zeta_k(u,v;\boldsymbol{x})\right. \\ &+ \left.\frac{1}{\zeta_k(u,v;\boldsymbol{x})}\frac{\partial^2}{\partial x_h \partial x_r}\zeta_k(u,v;\boldsymbol{x})\right).\end{aligned} \tag{5.76}$$

where

$$\begin{aligned}\frac{\partial^2}{\partial x_h \partial x_r}\zeta_k(u,v;\boldsymbol{x}) &= \sum_{i=1}^{K}\frac{\partial^2}{\partial x_h \partial x_r}\beta_i(v;\boldsymbol{x})p(q_v=i|q_u=k,\boldsymbol{x}) + \frac{\partial}{\partial x_h}\beta_i(v;\boldsymbol{x})\frac{\partial}{\partial x_r}p(q_v=i|q_u=k,\boldsymbol{x}) \\ &+ \frac{\partial}{\partial x_h}\beta_i(v;\boldsymbol{x})\frac{\partial}{\partial x_r}p(q_v=i|q_u=k,\boldsymbol{x}) + \beta_i(v;\boldsymbol{x})\frac{\partial^2}{\partial x_h \partial x_r}p(q_v=i|q_u=k,\boldsymbol{x}).\end{aligned} \tag{5.77}$$

- Final step:

$$\begin{aligned}\frac{\partial^2}{\partial x_h \partial x_r}p(\boldsymbol{y}|\boldsymbol{x}) &= \sum_{k=1}^{K}\left\{\frac{\partial^2}{\partial x_h \partial x_r}\beta_k(1;\boldsymbol{x})\right\}p(q_1=k|\boldsymbol{x}) + \left\{\frac{\partial}{\partial x_h}\beta_k(1;\boldsymbol{x})\right\}\left\{\frac{\partial}{\partial x_r}p(q_1=k|\boldsymbol{x})\right\} \\ &+ \left\{\frac{\partial}{\partial x_r}\beta_k(1;\boldsymbol{x})\right\}\left\{\frac{\partial}{\partial x_h}p(q_1=k|\boldsymbol{x})\right\} + \beta_k(1;\boldsymbol{x})\left\{\frac{\partial^2}{\partial x_h \partial x_r}p(q_1=k|\boldsymbol{x})\right\}.\end{aligned} \tag{5.78}$$

Finally, we need the derivatives of the log-likelihood:

$$\frac{\partial^2}{\partial x_h \partial x_r} \log p(\boldsymbol{y}|\boldsymbol{x}) = \frac{p(\boldsymbol{y}|\boldsymbol{x}) \frac{\partial^2}{\partial x_h \partial x_r} p(\boldsymbol{y}|\boldsymbol{x}) - \frac{\partial}{\partial x_h} p(\boldsymbol{y}|\boldsymbol{x}) \frac{\partial}{\partial x_r} p(\boldsymbol{y}|\boldsymbol{x})}{(p(\boldsymbol{y}|\boldsymbol{x}))^2}. \tag{5.79}$$

The elements of the observed Fisher information matrix are calculated given a set of trees $\mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ sampled by model $p(\cdot|\boldsymbol{x})$:

$$\hat{\mathbf{F}}(\boldsymbol{x})_{h,r} = -\frac{1}{N} \sum_{n=1}^{N} \frac{\partial^2}{\partial x_h \partial x_r} \log p(\boldsymbol{y}^{(n)}|\boldsymbol{x}). \tag{5.80}$$

The above calculations depend on the 2-nd order derivatives of initial state, state transition and state-conditional emission probabilities with respect to the latent coordinates:

- 2-nd order derivatives for initial probability for state k under the parametrisation in (4.18):

$$\begin{aligned}
\frac{\partial^2}{\partial x_h \partial x_r} p(q_1 = k|\boldsymbol{x}) \quad = \quad & \frac{\partial}{\partial x_h} g_k(\boldsymbol{W}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x})) \Bigg(\boldsymbol{W}^{(\boldsymbol{\pi})} \frac{\partial}{\partial x_r} \phi(\boldsymbol{x}) - \sum_{i=1}^{K} g_i(\boldsymbol{W}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{W}_i^{(\boldsymbol{\pi})} \frac{\partial}{\partial x_r} \phi(\boldsymbol{x}) \Bigg) \\
& + g_k(\boldsymbol{W}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x})) \Bigg(\boldsymbol{W}_k^{(\boldsymbol{\pi})} \frac{\partial^2}{\partial x_h \partial x_r} \phi(\boldsymbol{x}) - \sum_{i=1}^{K} \Bigg[\frac{\partial}{\partial x_h} g_i(\boldsymbol{W}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{W}_i^{(\boldsymbol{\pi})} \frac{\partial}{\partial x_r} \phi(\boldsymbol{x}) \\
& + g_i(\boldsymbol{W}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{W}_i^{(\boldsymbol{\pi})} \frac{\partial^2}{\partial x_h \partial x_r} \phi(\boldsymbol{x}) \Bigg] \Bigg).
\end{aligned} \tag{5.81}$$

- 2-nd order derivatives for transition probability from state k to state l under the parametrisation in (4.19):

$$\begin{aligned}
& \frac{\partial^2}{\partial x_h \partial x_r} p(q_u = l|q_{\rho(u)} = k, \boldsymbol{x}) = \\
& \frac{\partial}{\partial x_h} g_l(\boldsymbol{W}^{(\boldsymbol{B}_k)} \boldsymbol{\phi}(\boldsymbol{x})) \Bigg(\boldsymbol{W}_l^{(\boldsymbol{B}_k)} \frac{\partial}{\partial x_r} \phi(\boldsymbol{x}) - \sum_{i=1}^{K} g_i(\boldsymbol{W}^{(\boldsymbol{B}_k)} \boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{W}_i^{(\boldsymbol{B}_k)} \frac{\partial}{\partial x_r} \phi(\boldsymbol{x}) \Bigg) \\
& + g_l(\boldsymbol{W}^{(\boldsymbol{B}_k)} \boldsymbol{\phi}(\boldsymbol{x})) \Bigg(\boldsymbol{W}_l^{(\boldsymbol{B}_k)} \frac{\partial^2}{\partial x_h \partial x_r} \phi(\boldsymbol{x}) - \sum_{i=1}^{K} \Bigg[\frac{\partial}{\partial x_h} g_i(\boldsymbol{W}^{(\boldsymbol{B}_k)} \boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{W}_i^{(\boldsymbol{B}_k)} \frac{\partial}{\partial x_r} \phi(\boldsymbol{x}) \\
& + g_i(\boldsymbol{W}^{(\boldsymbol{B}_k)} \boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{W}_i^{(\boldsymbol{B}_k)} \frac{\partial^2}{\partial x_h \partial x_r} \phi(\boldsymbol{x}) \Bigg] \Bigg).
\end{aligned} \tag{5.82}$$

- 2-nd order derivatives for means of the emission distribution for state k under the parametrisation (4.17):

$$\frac{\partial^2}{\partial x_h \partial x_r} \boldsymbol{\mu}_k = \frac{\partial^2}{\partial x_h \partial x_r} \boldsymbol{W}^{(\boldsymbol{\psi}_k)} \phi(\boldsymbol{x}) = \boldsymbol{W}^{(\boldsymbol{\psi}_k)} \frac{\partial^2}{\partial x_h \partial x_r} \phi(\boldsymbol{x}). \tag{5.83}$$

The above derivatives regarding the initial state, state transition and emission parameters are calculated in the same fashion as the corresponding 2-st order derivatives in the case of GTM-HMM (Appendix D). The 2-nd order derivatives of the basis function $\frac{\partial^2}{\partial x_h \partial x_r}\phi(\boldsymbol{x})$ are identical to (5.63).

5.6.4 FIM for GTM-MTM

The same procedure for calculating magnification factors for GTM-HMM and GTM-HMTM is re-iterated here in the setting of GTM-MTM. Points $\boldsymbol{x}$ in latent space $\mathcal{V}$ are mapped to a two-dimensional manifold $\mathcal{M}$ of MTMs. In order to appreciate the magnification around a point $\boldsymbol{x}$, we need to estimate the local Fisher information matrix. This requires the 1-st and 2-nd order derivatives of the model log-likelihood of $p(\cdot|\boldsymbol{x})$ with respect to the coordinates of $\boldsymbol{x}$. It also requires the 1-st and 2-nd order derivatives of the MTM parameters with respect to the coordinates of $\boldsymbol{x}$. Moreover, as required by the Fisher information matrix in (5.31), likelihood is evaluated over a set of sample trees $\mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ generated by $p(\cdot|\boldsymbol{x})$. First, the 1-st order derivatives are presented followed by the 2-nd order derivatives.

1-st Order Derivatives

The log-likelihood for MTMs is calculated in section 3.4.4 via (3.97). We restate it here for $p(\cdot|\boldsymbol{x})$ addressed by $\boldsymbol{x}$ and calculated on a single tree $\boldsymbol{y}$:

$$\log p(\boldsymbol{y}|\boldsymbol{x}) \quad = \quad \sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{r=1}^{R} \nu_{rkl} \log p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}), \tag{5.84}$$

We calculate the 1-st order derivatives of the log-likelihood of MTM $p(\cdot|\boldsymbol{x})$ with respect to the latent coordinates x_1, x_2. Let $h \in \{1, 2\}$:

$$\begin{aligned}\frac{\partial}{\partial x_h}\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{r=1}^{R} \nu_{rkl} \log p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}) = \\ \sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{r=1}^{R} \nu_{rkl} \frac{1}{p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}})} \frac{\partial}{\partial x_h} p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}).\end{aligned} \tag{5.85}$$

The 1-st order derivative of transition parameter $p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r)$ under the

parametrisation in (4.38) is:

$$
\begin{aligned}
&\frac{\partial}{\partial x_h} p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}) = \\
&\frac{\partial}{\partial x_h} g_l(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x})) = \frac{\partial}{\partial x_h} \frac{\exp(\boldsymbol{W}_l^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))}{\sum_{j=1}^K \exp(\boldsymbol{W}_j^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))} \\
&= \frac{\exp(\boldsymbol{W}_l^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_l^{r,k}\frac{\partial}{\partial x_h}\boldsymbol{\phi}(\boldsymbol{x})\Big(\sum_{j=1}^K \exp(\boldsymbol{W}_j^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\Big)}{\Big(\sum_{j=1}^K \exp(\boldsymbol{W}_j^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\Big)^2} \\
&- \frac{\exp(\boldsymbol{W}_l^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\Big(\sum_{j=1}^K \exp(\boldsymbol{W}_j^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_l^{r,k}\frac{\partial}{\partial x_h}\boldsymbol{\phi}(\boldsymbol{x})\Big)}{\Big(\sum_{j=1}^K \exp(\boldsymbol{W}_j^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\Big)^2} \\
&= g_l(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\Big(\boldsymbol{W}_l^{r,k}\frac{\partial}{\partial x_h}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{j=1}^K g_j(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_l^{r,k}\frac{\partial}{\partial x_h}\boldsymbol{\phi}(\boldsymbol{x})\Big).
\end{aligned}
\tag{5.86}
$$

The 1-st order derivatives of the basis function $\frac{\partial}{\partial x_r}\phi(\boldsymbol{x})$ are identical to (5.55).

2-nd Order Derivatives

We proceed with the calculation of the 2-nd order derivatives of the log-likelihood of MTM $p(\cdot|\boldsymbol{x})$ via (5.84) with respect to the coordinates of $\boldsymbol{x}$. Let $j, h \in \{1, 2\}$:

$$
\begin{aligned}
&\frac{\partial^2}{\partial x_j \partial x_h} \sum_{k=1}^K \sum_{l=1}^K \sum_{r=1}^R \nu_{rkl} \log p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}) = \\
&\sum_{k=1}^K \sum_{l=1}^K \sum_{r=1}^R \nu_{rkl} \Big(\frac{-1}{(p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}))^2} \frac{\partial}{\partial x_j} p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}) \\
&\times \quad \frac{\partial}{\partial x_h} p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}) \\
&+ \quad \frac{1}{p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}})} \frac{\partial^2}{\partial x_j \partial x_h} p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}) \Big).
\end{aligned}
\tag{5.87}
$$

The 2-nd order derivative of transition parameter $p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r)$ under the parametrisation in (4.38) is:

$$\frac{\partial^2}{\partial x_j \partial x_h} p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r, \boldsymbol{\mathcal{W}}) =$$

$$\frac{\partial}{\partial x_j}\Bigg\{ g_l(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\Bigg(\boldsymbol{W}_l^{r,k}\frac{\partial}{\partial x_h}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{j=1}^{K} g_j(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_l^{r,k}\frac{\partial}{\partial x_h}\boldsymbol{\phi}(\boldsymbol{x})\Bigg)\Bigg\} =$$

$$\Bigg\{ \frac{\partial}{\partial x_j} g_l(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\Bigg\}\Bigg(\boldsymbol{W}_l^{r,k}\frac{\partial}{\partial x_h}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{j=1}^{K} g_j(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_l^{r,k}\frac{\partial}{\partial x_h}\boldsymbol{\phi}(\boldsymbol{x})\Bigg)$$

$$+ \quad g_l(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\Bigg(\boldsymbol{W}_l^{r,k}\frac{\partial^2}{\partial x_j \partial x_h}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{j=1}^{K} \frac{\partial}{\partial x_j} g_j(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_l^{r,k}\frac{\partial}{\partial x_h}\boldsymbol{\phi}(\boldsymbol{x})$$

$$+ \quad g_j(\boldsymbol{W}^{r,k}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{W}_l^{r,k}\frac{\partial^2}{\partial x_j \partial x_h}\boldsymbol{\phi}(\boldsymbol{x})\Bigg). \tag{5.88}$$

The 2-nd order derivatives of the basis function $\frac{\partial^2}{\partial x_h \partial x_r}\phi(\boldsymbol{x})$ are identical to (5.63).

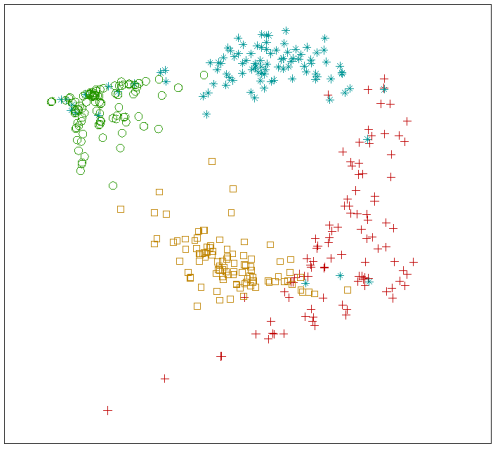
The elements of the observed Fisher information matrix are calculated given a set of trees $\mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ that are sampled by model $p(\cdot|\boldsymbol{x})$ are given in the same fashion as for GTM-HMTM in section 5.6.3:

$$\hat{\mathbf{F}}(\boldsymbol{x})_{h,r} = -\frac{1}{N}\sum_{n=1}^{N} \frac{\partial^2}{\partial x_h \partial x_r} \log p(\boldsymbol{y}^{(n)}|\boldsymbol{x}). \tag{5.89}$$

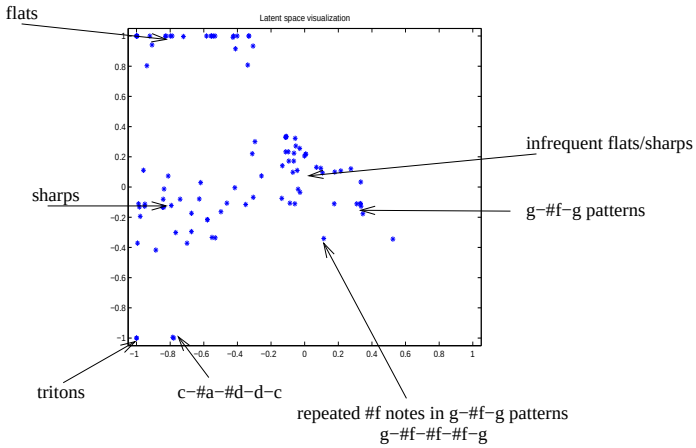
5.7 Experiments and Results on Magnification Factors

In this section we experimentally illustrate the KLD approximation and the FIM method in revealing the magnification factors of the manifold $\mathcal{M}$ of local noise models. In order to illustrate the magnification factors on manifold $\mathcal{M}$, for each latent centre $\boldsymbol{x}_c, c = 1, 2, \ldots, C$, we compute the KLD between each $\boldsymbol{x}_c$ and its perturbation $\boldsymbol{x}_c + \boldsymbol{dx}$ by using the equations presented in the preceding sections. Here we define a rectangular, regular grid of 25×25 points on the latent space. This divides the grid in 625 squares with a latent point at the centre of each square. Increasing the number of grid points results to finer magnification plots. As previously described, we perturb each latent centre $\boldsymbol{x}$ in 16 regularly spaced directions on a small circle (we have set its radius to 10^{-5}). This is illustrated in Fig. 5.2. The same procedure is employed for calculating the magnification factors around each $\boldsymbol{x}_c$ via FIM. In the case of FIM, we note that alternatively, we could have used singular value decomposition of the FIM to find and quantify the local dominant stretching directions in the latent space.

For each latent point, out of the 16 directions of perturbation, the direction of maximal magnification is represented by a straight line drawn through the centre of the corresponding



(a)



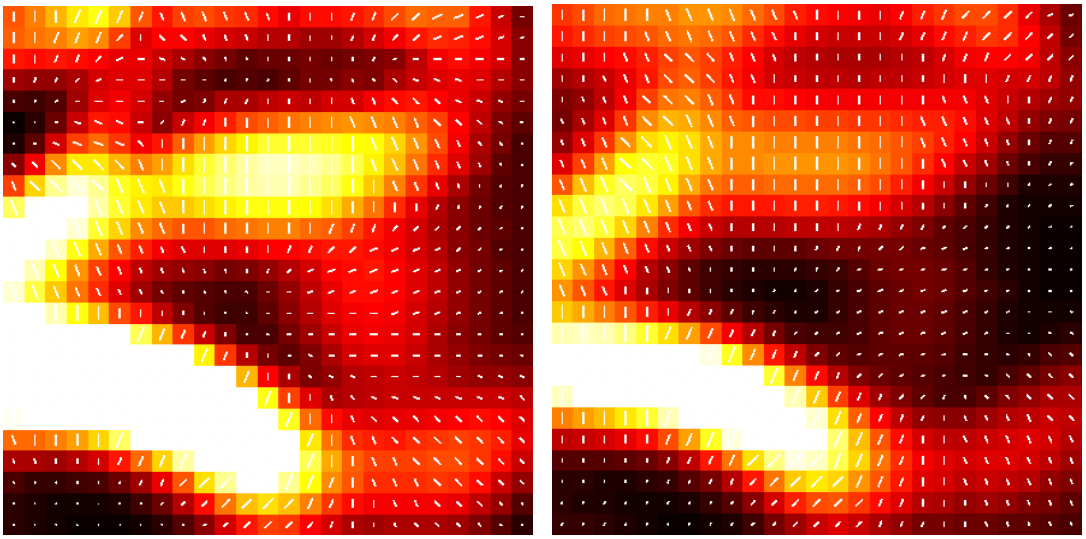
(b)

Fig. 5.6: Visualisation of toy dataset of binary sequences (a) and Bach chorals [Tiňo et al., 2004] (b) via GTM-HMM.

square. The length of the line signifies the level of the magnification, i.e. shorter lines indicate lower magnification, longer lines indicate higher magnification. The plots are coloured as heat maps. Colours rank from white (highest values) to yellow to red to black/dark (lowest values) and signify the level of magnification. Thus, white squares are associated with high magnification, whereas black squares are associated with low magnification.

5.7.1 Hidden Markov Models

As aforementioned, the GTM-HMM is introduced in [Tiňo et al., 2004] as an extension of GTM to sequences. We illustrate the magnification factors for GTM-HMM on two datasets. The first dataset is a toy dataset constructed by sampling 100 binary sequences of length 40 from four HMMs ($K = 2$ hidden states) with identical emission structure, i.e. the HMMs differed



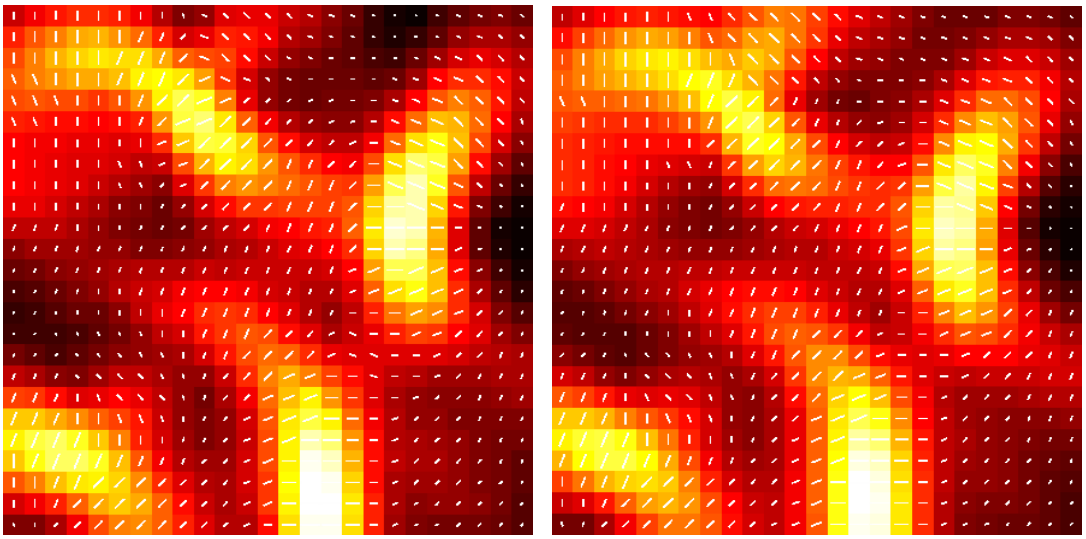
(a) (b)

Fig. 5.7: Magnification factors via FIM (a) and KLD (b) for GTM-HMM on toy dataset.

only in transition probabilities. The other dataset is a set of 100 chorales by J.S. Bach from the UCI repository of Machine Learning Databases and was used as one of the datasets in the experiments in [Tiňo et al., 2004]. GTM-HMM was trained on both datasets and produced the visualisation plots displayed in Fig. 5.6(a) and 5.6(b). In Fig. 5.6(a) toy sequences are marked with four different markers, corresponding to the four different HMMs used to generate the data set. We stress that the model was trained in a completely unsupervised way and that the markers are used for illustrative purposes only. The four clusters are clearly discerned. Of course, GTM-HMM benefits from the fact that the distributions used to generate data were from the same model class as the local noise models. In Fig. 5.6(b) GTM-HMM has organised the Bach choral sequences in a fashion that reflects certain melodic motives. Three main regions are discerned, one in the upper part of the plot where flats dominate, the central left where sharps dominate and the centre of the plot where we find that sharps and flats are very rare.

We used the trained GTM-HMM on the toy dataset to obtain representations of the induced metric in the local noise model space based on the FIM and KLD approximation which can be seen in Fig. 5.7(a) and 5.7(b), respectively. The plots are practically identical, which means that the KLD approximation is accurate. The bright boundaries (white to yellow) indicate clear separations on the map, and we identify four dark regions, one at the top left, one at the top centre, one at the centre separated by a not a pronounced boundary from another dark region at the right centre of the plot. These regions seem to correspond to the discovered clusters.

Fig. 5.8(a) and 5.8(b) illustrate the magnification factors for GTM-HMM trained on the Bach chorals via Fisher information matrix and KLD approximation respectively. Here again



(a) (b)

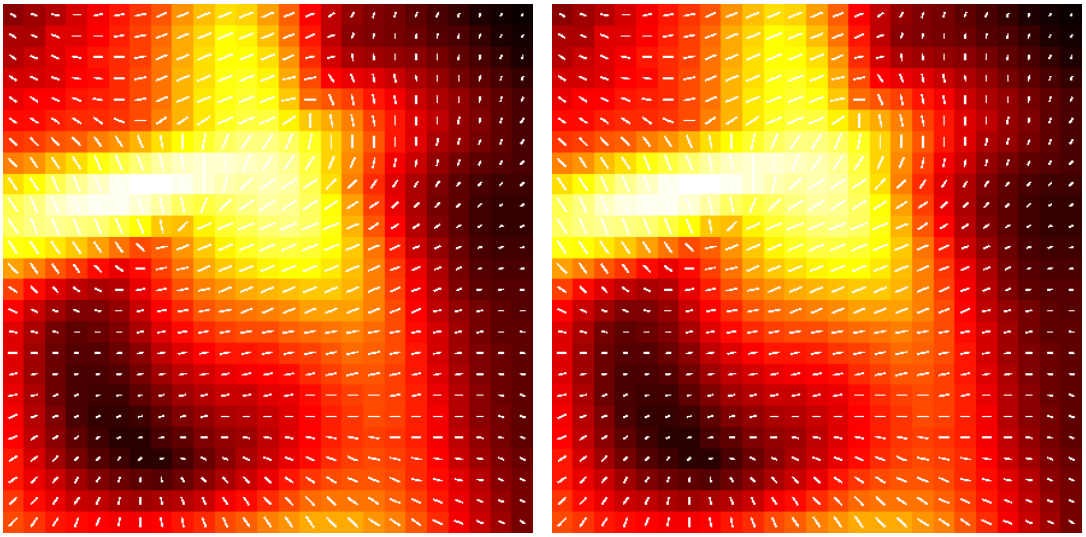
Fig. 5.8: Magnification factors via FIM (a) and KLD (b) for GTM-HMM on Bach chorals dataset.

the KLD and FIM plots are practically identical. The magnification plots appear more complex in this case, as there exist certain formations in locations where no data points are projected. These formations are artifacts that do not model any subset of the data, but appear due to the foldings of the map in the higher dimensional space as it attempts to capture the topographic structure of the data. However, in the three main aforementioned locations in Fig. 5.6(b) where data points do reside, dark regions are visible denoting the presence of data groups. Thus we see that the upper region is clearly separated and that in the centre two mildly separated dark regions are present. At the bottom left corner of the plot we see a small well separated dark region that seems to correspond to the few sequences mapped in the same location in plot Fig. 5.6(b). These sequences seem to differ significantly when compared to the rest.

5.7.2 Hidden Markov Tree Models

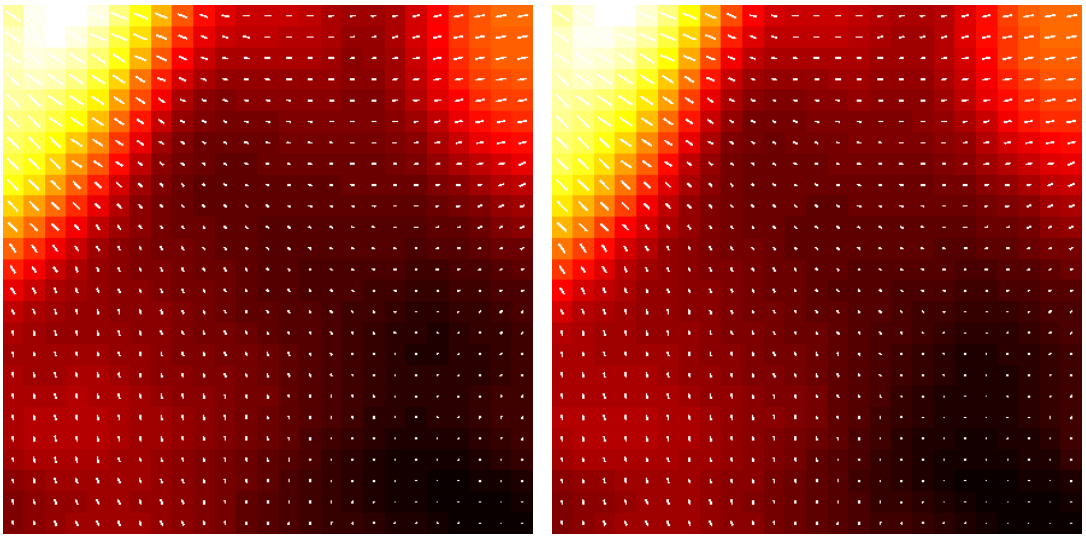
We calculated magnification factors on all three datasets of section 4.3.1, the toy dataset created by sampling HMTMs, the TPB, and the quadtree dataset. Regarding the toy dataset, we observe that both plots of Fig. 5.9(a) and 5.9(b) illustrate very similar magnifications for the toy dataset. It can be seen in both figures that the data have been organised in 4 distinct clusters, well separated by bright regions that signify that the clusters have clear boundaries and are indeed different from each other (compare with Fig. 4.6(a)).

Fig. 5.10(a) and 5.10(b) illustrate the magnification plots for the TPB dataset. The bright region concentrated in the left upper corner of the plot concerns the topographic organisation of



(a) (b)

Fig. 5.9: Magnification factors via FIM (a) and KLD approximation (b) for GTM-HMTM on toy dataset.



(a) (b)

Fig. 5.10: Magnification factors via FIM (a) and KLD approximation (b) for GTM-HMTM on TPB dataset.

the two ship classes (see Fig. 4.6(b)). The high magnification indicates that data points projected in this area are very dissimilar to each other as abrupt changes occur in the underlying models. This is verified by the fact that we have identified that class $*$, the class of ships with two masts, has been split into three sub-clusters. On the other hand, the classes of policemen exhibit a gentler separation between them as indicated by the moderately light area close to the right upper corner. Finally the region of the classes of houses, which as we saw earlier have all been

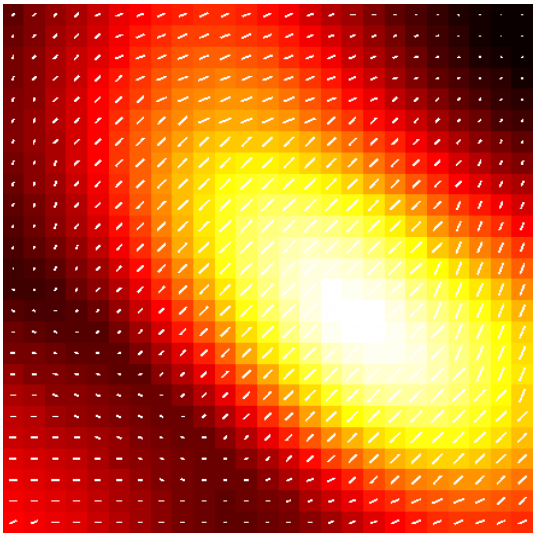


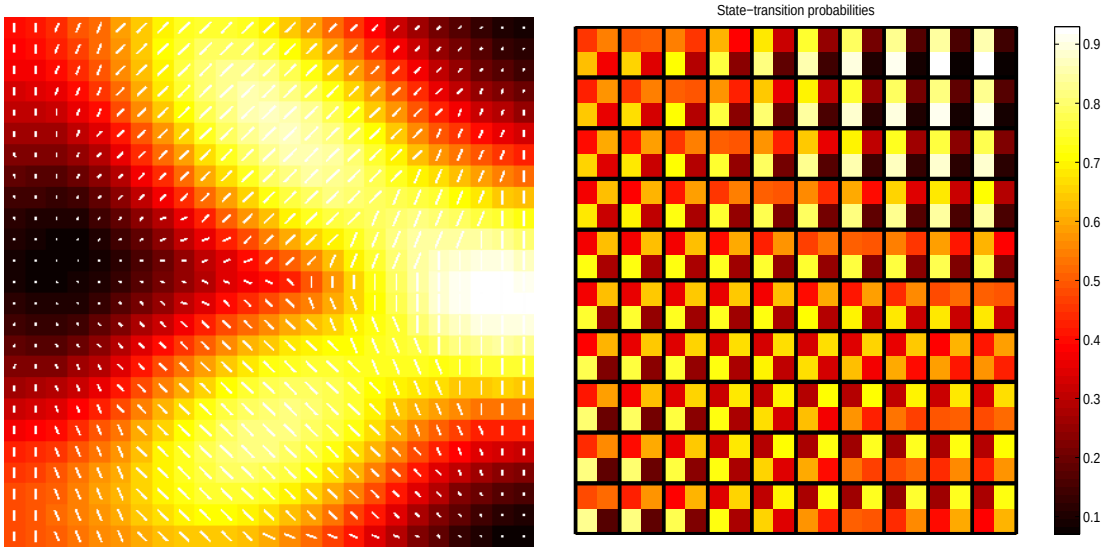
Fig. 5.11: KLD approximation for GTM-HMTM on quadtree dataset.

projected in one super-cluster, is characterised by very low magnification. This suggests, that indeed this super-cluster is dense and that the underlying models fail to discern differences between house patterns.

In Fig. 5.11 magnification factors for the quadtree dataset are displayed using the KLD approximation method. The plot does not impart information on the presence of any clusters. However, when inspected in conjunction with the state transitions in Fig. 4.10 and the means of the emissions in Fig. 4.11, we can see how it reflects the situation of the visualisation plot in Fig. 4.9 where an overlapping of images of different orientations occurs. We recall that transition probabilities vary only little across the plot. Also, the plots of means exhibit a common abrupt change close to their respective centres. We see that the magnification factors effectively summarise the behaviour of the means corresponding to the three states: a high magnification is observed at the centre of the plot where the means change the most. The magnification factor plots via FIM were virtually identical to the ones obtained via KLD approximation.

5.7.3 Markov Tree Models

The magnification factors for the toy dataset of section 4.6 are presented in Fig. 5.12(a). The clusters illustrated in Fig. 4.15 are visible here too, clearly separated by bright boundaries signifying stretches in the manifold of local noise models. Inspecting the state transitions for the toy dataset (similar to Fig. 4.7), a clear structure is observed. For example in Fig. 5.12(b), where the transition probabilities that correspond to the 3-rd child are illustrated, the regions underlying the three clusters indicate different trends. In the case of quadtree data set, the *local*



(a) (b)

Fig. 5.12: Magnification factors for GTM-MTM on toy dataset (a). State transitions for 3-rd child node for toy dataset (b).

metric structure of GTM-MTM was varying rather slowly and so the magnification factor plot (reflecting local differentiable structure of the noise manifold) was almost flat. The topographic organisation is driven by small local changes in the noise models. Maps of magnification factors are not well suited for such situations.

We also calculated magnification factors via FIM. The magnification factor plots were virtually identical to the ones obtained via KLD approximation.

Chapter 6

An Extension of GTM to the Visualisation of Astronomical Light Curves

Binary stars are gravitationally bound pairs of stars that orbit a common centre of mass. Astronomical observations suggest that almost half of all stars are binary ones [Guinan and Engle, 2006]. Thus, study of such systems procures knowledge for a significant proportion of stars. Amongst other reasons, binary stars are important to astrophysics because they allow calculation of fundamental quantities such as masses and radii. The increasing number of binary star discoveries, provides samples for the testability of theoretical models for stellar formation and evolution. Also, by measuring their fundamental parameters they can serve as distance indicators to galaxies. Moreover, the study of binaries has led to the discovery of extrasolar planets. A particular subclass of binary stars are eclipsing binary stars. The luminosity of such stars varies over time and forms a graph called a light curve. Light curves are important because they provide information on the characteristics of stars and help in the identification of their type.

In this chapter we develop a novel GTM extension for the construction of topographic maps of light curves of eclipsing binary stars. To that purpose we need to formulate an appropriate noise model. Here the noise model is a physical model to which Gaussian noise is added.

6.1 Light Curve Model

6.1.1 Physical Model

The physical model that generates light curves from eclipsing binary systems is described by the following set of parameters: mass $M_1 \in [0.5, 100]$ (in solar masses) of the primary star (star with highest mass of the pair), mass ratio $q \in [0, 1]$ (hence mass of secondary star is $M_2 = qM_1$), eccentricity $e \in [0, 1]$ of the orbit and period $\rho \in [0.5, 100]$ measured in days, all of which specify the shape of the orbit. Furthermore, three angles describing the orientation of the system are necessary [Hilditch, 2001]. Perpendicular to the line of sight of the observer is a plane of reference called the plane of the sky (see Fig. 6.1). The plane containing the orbit of the binary system is called the orbital plane. The angle between the plane of the sky and the orbit plane is called inclination $\imath \in [0, \frac{\pi}{2}]$. Thus, an angle of $\imath = \frac{\pi}{2}$ signifies an edge on view of the system. The plane of the sky and the orbit intersect at two points N and N'. The line connecting points N, N' is called the line of nodes. The angle that orients the orbital plane with respect to the line of nodes is called the longitude of ascending node $\Omega \in [0, 2\pi]$ (and is measured on the plane of sky). Angles $\imath$ and Ω together orient the orbital plane with respect to the plane of the sky. Finally, the argument of periastron is the angle $\omega \in [0, 2\pi]$ that orients the major axis of the elliptic orbit within its plane, that is ω is measured within the orbital plane. These angles are illustrated in Fig. 6.1. However, angle Ω has no effect on the observed light curves and is omitted from the model.

The mass of each star is linked to the luminosity radiated by a surface element of the star according to the mass-luminosity relation:

$$L = M^{3.5}. \tag{6.1}$$

Moreover, masses are related to the radii of the stars according to the mass-radius relation:

$$R = \begin{cases} 10^{0.053+0.977\log_{10}(M)}, & \text{if } M < 1.728; \\ 10^{0.153+0.556\log_{10}(M)}, & \text{otherwise.} \end{cases} \tag{6.2}$$

From these relations we see that the primary star is the most luminous one and the one with the greatest area of the pair (a star appears as a disc to an observer). Thus, the observed area of a star is $A = \pi R^2$ and the observed luminosity is $L\pi R^2$. Henceforth, we shall index quantities related to the stars of a binary system by 1 for the primary star (e.g. primary mass is M_1) and 2 for the secondary star.

It can be proved from Newton's laws (amongst other methods also with pure geometrical arguments [Goodstein and Goodstein, 1996]) that the orbits of one object in the gravitational field of the other object is a conic section of eccentricity e. Values of $e = 1$ and $e > 1$ correspond respectively to a parabola and hyperbola which are not closed orbits. Here we are interested in the case where $0 \leq e < 1$. Furthermore, a two-body system can be equivalently formulated as a system of two bodies where one is fixed and only one is in orbital motion. This formulation is useful in the case we are interested in the relative motion of one of the bodies, as we are in this case, where we are concerned with light curves. It is shown in [Hilditch, 2001] that in the relative motion system, the eccentricity, period and semi-major of the moving body's orbit are equal to their counterparts in the two-body system, and only the masses transform.

The position of the orbiting body is calculated by Kepler's equation as the distance r from the fixed companion star on the elliptical orbit [Hilditch, 2001].

$$r(t) = \frac{a(1-e^2)}{1 + e\cos\theta(t)}, \tag{6.3}$$

where t is time and a is the semi-major axis of the ellipse calculated by Kepler's third law. Point Π in Fig. 6.1 is the periastron, the point where the distance between the orbiting and fixed body is minimum. Angle θ is the angle between the radius and the periastron. Knowledge of θ would allow us to determine the position of the orbiting body. Angle θ is indirectly inferred via an auxiliary circle centred at the centre of the ellipse O and radius equal to semi-major axis. Point Q is the vertical projection of the orbiting body's position P to the auxiliary circle. Angle E is called the eccentric anomaly and is given by Kepler's equation [Hilditch, 2001]:

$$E(t) = e\sin E(t) + \frac{2\pi}{\rho}(t-\tau), \tag{6.4}$$

where τ is the instance of time that the body was at the periastron. Kepler's equation does not admit an analytical solution. Instead an approximate solution may be found through the Newton-Raphson method. The iteration starts with a first approximation of $E_0 = \frac{2\pi}{\rho}(t-\tau)$ and is repeated until convergence:

$$\begin{aligned} E_0(t) &= \frac{2\pi}{\rho}(t-\tau), \\ E_i(t) &= \frac{2\pi}{\rho}(t-\tau) + e\sin E_{i-1}(t). \end{aligned} \tag{6.5}$$

Via geometrical considerations it is proved that the relation between the true and eccentric

anomaly is:

$$\tan\frac{\theta(t)}{2} = [(1+e)/(1-e)]^{\frac{1}{2}}\tan(\frac{E(t)}{2}) \tag{6.6}$$

By knowledge of θ we can fix the position of the second star on the orbit using (6.3) and (6.6). These positions correspond to the orbital plane and must be projected to the plane of the observer in the form of Cartesian coordinates [Hilditch, 2001]:

$$\begin{aligned} X(t) &= r(t)(\cos(\Omega)\cos(\omega+\theta(t)) - \sin(\Omega)\sin(\omega+\theta(t))\cos(\imath)), & (6.7)\\ Y(t) &= r(t)(\cos(\Omega)\cos(\omega+\theta(t)) + \cos(\Omega)\sin(\omega+\theta(t))\cos(\imath)), & (6.8)\\ Z(t) &= r(t)\sin(\omega+\theta(t))\sin(\imath), & (6.9) \end{aligned}$$

which concludes the determination[1] of positions of the stars with respect to the observer.

An observer of the binary system receives a variable luminosity from the eclipsing binary system that plotted against time forms a light curve. This variability is due to the eclipses that occur when one body passes in front (in the line of sight of the observer) of the other. This is illustrated in Fig. 6.2. When no eclipse occurs (positions a, g) the luminosity is equal to the sum of the luminosities radiated from the two bodies. The curved parts of the light curve occur when a body partially obscures the other. We distinguish between partial eclipses (positions b, f, h, l) where the phenomenon of the eclipse is at its beginning, and deep eclipses (positions c, e, i, k) where the phenomenon is in a more advanced state. Two eclipses take place at each period, one primary eclipse (position d) which occurs when the most luminous body of the pair is obscured the most, and a secondary eclipse (position j) which occurs when the most luminous body obscures its companion the most.

Obscured parts of the disks of the stars can be calculated via geometrical arguments in the spirit of a simple light curve model in available[2] at http://www.physics.sfasu.edu/astro/ebstar/ebstar.html. The obscured area of each star is denoted by $\Delta A_1(t)$ and $\Delta A_2(t)$ at time t. The areas of the visible parts of the discs of the stars are summarised in table 6.1. More details on the derivation of the areas can be found in appendix F.

The luminosity received by the observer is:

$$f_{\boldsymbol{\theta}}(t) = L_1(A_1 - \Delta A_1(t)) + L_2(A_2 - \Delta A_2(t)), \tag{6.10}$$

where we introduce the notation $f_{\boldsymbol{\theta}}(t)$ that stands for the physical models of light curves

[1] Angle Ω does influence the position of the orbiting body. However, it does not have an influence on the light curve and thus we treat it as a constant $\Omega = 0$.

[2] Last accessed on the 12th September 2007.

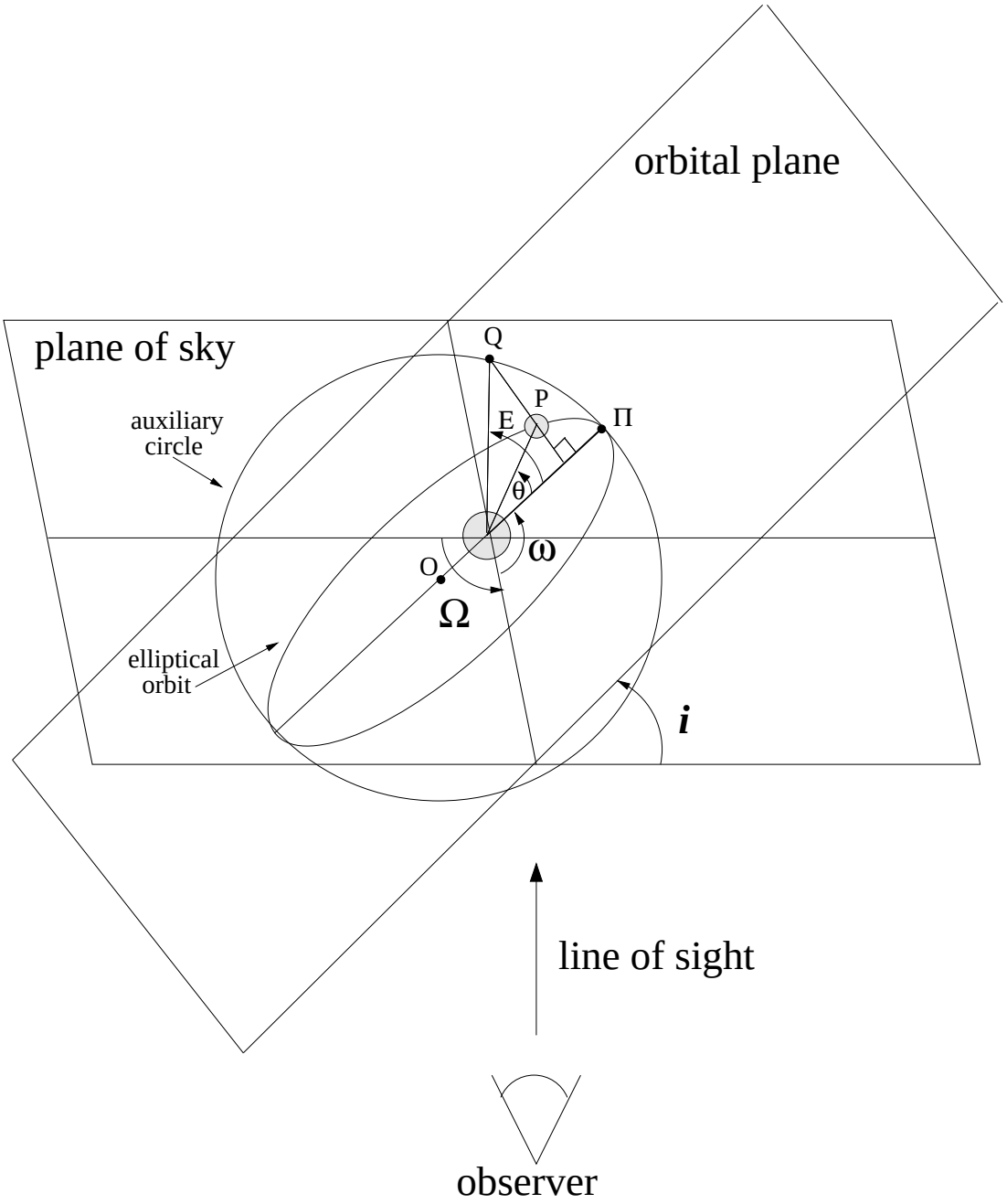


Fig. 6.1: Angles orientating the orbital plane with respect to the plane of sky. Adapted from [Hilditch, 2001].

presented here. Vector $\boldsymbol{\theta} = \{M_1, q, e, \imath, \omega, \rho\}$ is a vector of the parameters of the physical model. Having concluded an exposition of the light curve model, we note that the model is further enhanced with limb-darkening. Limb-darkening [Hilditch, 2001] refers to the phenomenon of the disc of the stars not being uniformly illuminated. Instead, a decrease in luminosity is observed from the centre of the discs towards the edges of the discs. However, we shall not delve

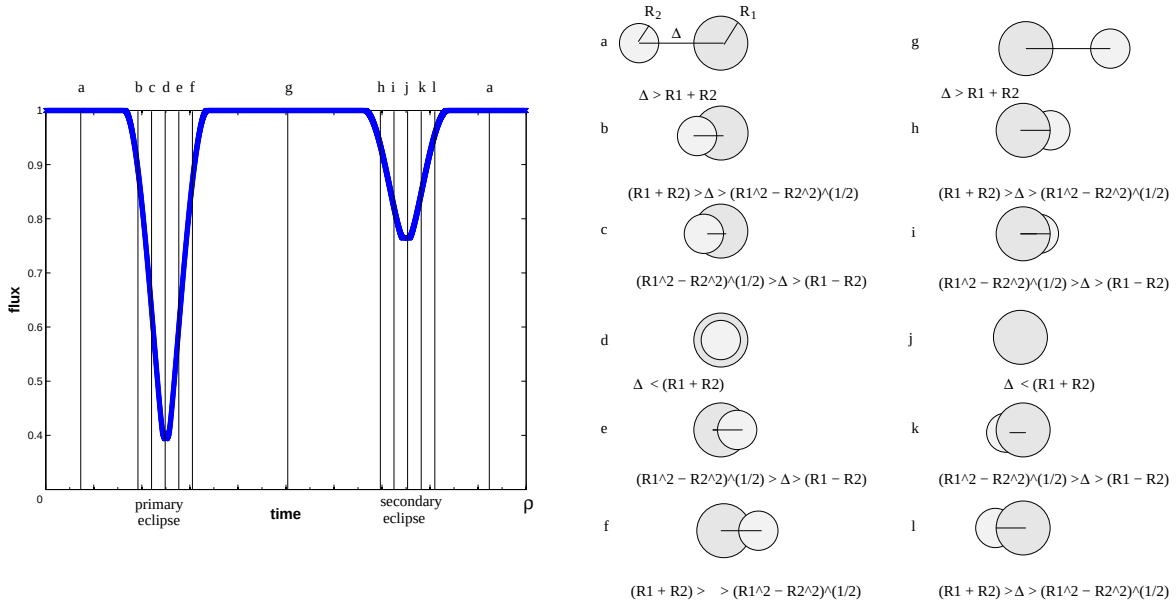


Fig. 6.2: Positions of stars (relative to observer's line of sight) and corresponding light curve phases. Δ is the separation of the centres of gravity of the stars.

Table 6.1: Summary the areas of the visible discs of the stars. Column 'pos' refers to positions in Fig. 6.2. Taken from http://www.physics.sfasu.edu/astro/ebstar/ebstar.html.

		Primary star in front		**Secondary star in front**	
state	**pos**	$\boldsymbol{A_1}$	$\boldsymbol{A_2}$	$\boldsymbol{A_1}$	$\boldsymbol{A_2}$
no eclipse	a,g	πR_1^2	πR_2^2	πR_1^2	πR_2^2
partial eclipse	b,f,h,l	πR_1^2	$\pi R_2^2 - \Delta A_1 - \Delta A_2$	$\pi R_1^2 - \Delta A_2 - \Delta A_1$	πR_2^2
deep eclipse	c,e,i,k	πR_1^2	$\Delta A_1 - \Delta A_2$	$\pi R_1^2 - \pi R_2^2 + \Delta A_2 - \Delta A_1$	πR_2^2
eclipse	d,j	πR_1^2	0	$\pi R_1^2 - \pi R_2^2$	πR_2^2

into further detail regarding this phenomenon.

6.1.2 Prior Distribution on Model Parameters

The light curve model is supplemented by a set of prior densities on the physical parameters $\{M_1, q, e, \imath, \omega, \rho\}$ of the model that have been obtained from relevant literature. Details on the priors can be found in Appendix E.

6.1.3 Generative Noise Model for Light Curves

Based on the physical model a probabilistic generative noise model arises naturally. Observed light curves, denoted by $\boldsymbol{O}$, are noisy signals:

$$\boldsymbol{O}(t) = f_{\boldsymbol{\theta}}(t) + \epsilon(t), \tag{6.11}$$

where ϵ is independent and identically distributed Gaussian noise with variance σ^2. Thus, we regard a light curve $\boldsymbol{O}$ as the emission of a multivariate homoscedastic normal distribution:

$$\boldsymbol{O} \sim \prod_{t=0}^{\rho(\mathbf{O})} \mathcal{N}(f_{\boldsymbol{\theta}}(t), \sigma^2), \tag{6.12}$$

where $\rho(\mathbf{O})$ is the period associated with $\boldsymbol{O}$.

Thus, in order to generate an observed light curve $\boldsymbol{O}$ the noise model simulates the physical system, identified by parameter vector $\boldsymbol{\theta}$, to produce a theoretical light curve and subsequently adds noise to it. We denote the noise model associated with parameters $\boldsymbol{\theta}$ by $p(\boldsymbol{O}|f_{\boldsymbol{\theta}}(.), \sigma^2)$ or simply by $p(\boldsymbol{O}|\boldsymbol{\theta})$. Thus, the probability that the physical system identified by $\boldsymbol{\theta}$ generates light curve $\boldsymbol{O}$ is calculated by:

$$p(\boldsymbol{O}|\boldsymbol{\theta}) = \prod_{t=0}^{\rho(\mathbf{O})} \mathcal{N}(\boldsymbol{O}(t), f_{\boldsymbol{\theta}}(t), \sigma^2). \tag{6.13}$$

In order to complete the noise model formulation $p(\boldsymbol{O}|\boldsymbol{\theta})$, a few matters require attention. Note that computing $p(\boldsymbol{O}|\boldsymbol{\theta})$ is problematic if the period of the observed light curve and the period of the noise model are not equal since time points t in (6.13) will exceed the time range of the light curve with shorter period. Also, it is appropriate when computing $p(\boldsymbol{O}|\boldsymbol{\theta})$ that compared points $\boldsymbol{O}(t)$ and $f_{\boldsymbol{\theta}}(t)$ correspond to the same phase. Two measures need to be taken:

- All light curves in dataset $\mathcal{D}$ are phase-shifted so that the first point in each light curve $\boldsymbol{O}$ corresponds to the point of its primary eclipse. We impose this requirement also to the noise model by phase-shifting its generated light curve $f_{\boldsymbol{\theta}}$ in the same way. This is illustrated in Fig. 6.3 where the light curve in Fig. 6.3(a) is shifted to yield the light curve in Fig. 6.3(b).

- Each light curve $\boldsymbol{O}$ is resampled so that its length becomes T. This is done by sampling $\boldsymbol{O}$ at T equally spaced points, with the first and last sampled points coinciding with the first and last points of $\boldsymbol{O}$ respectively. The resampled light curve replaces the original light curve $\boldsymbol{O}$ in dataset $\mathcal{D}$. We impose that the noise model also produces light curves $f_{\boldsymbol{\theta}}$

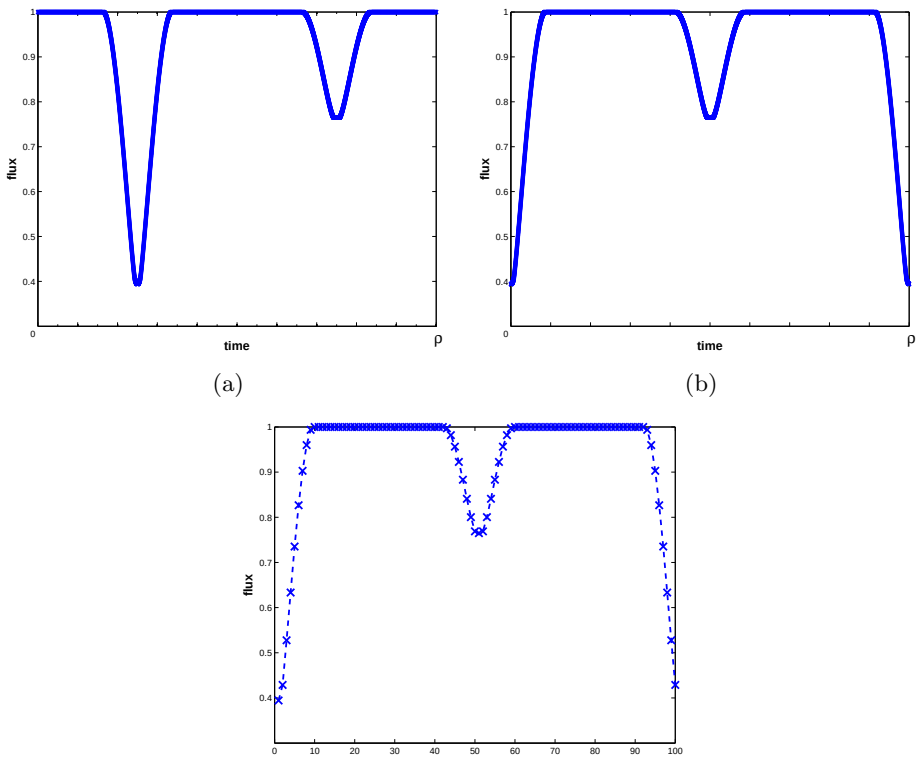


Fig. 6.3: Light curve in (a) is phase shifted so that its deepest point at the primary eclipse becomes the its starting point, which yields the light curve in (b). Light curve in (b) is then resampled so that it consists of T observations, which yields the light curve in (c).

of length T following the same resampling. This is illustrated in Fig. 6.3 where the light curve in Fig. 6.3(b) where has been sampled at T equally spaced points to yield the light curve in Fig. 6.3(c).

These two measures ensure that when comparing light curve $\boldsymbol{O}$ to light curve emitted by noise model $p(\boldsymbol{O}|\boldsymbol{\theta})$ in (6.13), the compared points points $\boldsymbol{O}(t)$ and $f_{\boldsymbol{\theta}}(t)$ correspond to the same phase.

6.2 Model for Topographic Organisation

The starting point of our model formulation is the form of a mixture model composed of C components:

$$p(\boldsymbol{O}|\boldsymbol{\Theta}) \quad = \quad \sum_{c=1}^{C} P(c)\; p(\boldsymbol{O}|\boldsymbol{\theta}_c), \tag{6.14}$$

where $P(c)$ are the mixing coefficients, $\boldsymbol{\Theta}$ is the set of all parameter vectors $\{\boldsymbol{\theta}_c\}_{c=1,\ldots,C}$ and $p(\boldsymbol{O}|\boldsymbol{\theta}_c)$ corresponds to the $c-$th model component with parameter vector $\boldsymbol{\theta}_c$. We simplify notation $p(\boldsymbol{O}|\boldsymbol{\theta}_c)$ to $p(\boldsymbol{O}|c)$. Assuming that the observations in $\mathcal{D}$ are independently generated, the likelihood $\mathcal{L}$ of the mixture model $p(\boldsymbol{O}|\boldsymbol{\Theta})$ is expressed as:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) &= p(\mathcal{D}|\boldsymbol{\Theta}) = \prod_{n=1}^{N} p(\boldsymbol{O}^{(n)}|\boldsymbol{\Theta}) = \prod_{n=1}^{N}\sum_{c=1}^{C} P(c)p(\boldsymbol{O}^{(n)}|c) \\ &\propto \prod_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{O}^{(n)}|c), \end{aligned} \tag{6.15}$$

where the mixing coefficients can be ignored as $P(c) = \frac{1}{C}$. Given the prior distributions on the physical parameters in section E, we perform MAP estimation. We express the posterior of $\boldsymbol{\Theta}$ as:

$$\begin{aligned} p(\boldsymbol{\Theta}|\mathcal{D}) &= \frac{p(\mathcal{D}|\boldsymbol{\Theta})p(\boldsymbol{\Theta})}{p(\mathcal{D})}, \\ \log p(\boldsymbol{\Theta}|\mathcal{D}) &= \log\frac{p(\mathcal{D}|\boldsymbol{\Theta})}{p(\mathcal{D})} + \log\frac{p(\boldsymbol{\Theta})}{p(\mathcal{D})} \\ &\propto \log p(\mathcal{D}|\boldsymbol{\Theta}) + \log p(\boldsymbol{\Theta}) \\ &\propto \log \mathcal{L}(\boldsymbol{\Theta}) + \log p(\boldsymbol{\Theta}) \\ &\propto \sum_{n=1}^{N}\log\sum_{c=1}^{C} p(\boldsymbol{O}^{(n)}|\boldsymbol{\theta}_c) + \log p(\boldsymbol{\Theta}), \end{aligned} \tag{6.16}$$

where $p(\boldsymbol{\Theta})$ is the prior probability term encompassing all priors $\boldsymbol{\theta}_c$ of each model component c. The prior probability term reads:

$$\begin{aligned}\log p(\boldsymbol{\Theta}) &= \sum_{c=1}^{C}\log p(\boldsymbol{\theta}_c) \\ &= \sum_{c=1}^{C}\Big(\log p(M_{1c}) + \log p(q_c) + \log p(e_c|\rho_c) + \log p(\omega_c) + \log p(i) + \log p(\rho_c)\Big). \end{aligned} \tag{6.17}$$

We would like to introduce a topographic organisation for the components of the mixture model. This can be achieved in the spirit of the GTM formulations in chapter 4. We require that each parameter vector $\boldsymbol{\theta}_c$ of component c is constrained on a regular grid of points $\boldsymbol{x}_c, c = 1,\ldots,C$ in a two dimensional space $[-1,1]^2$ that we denote by $\mathcal{V}$. A smooth nonlinear function Γ maps each point $\boldsymbol{x}$ in $\mathcal{V}$ to a point $\Gamma(\boldsymbol{x})$ that addresses a model $p(\cdot|\boldsymbol{x})$. Points $\Gamma(\boldsymbol{x})$ are constrained on a two-dimensional manifold $\mathcal{M}$ that is embedded in space $\mathcal{H}$, the space of

all models. Since the neighbourhood of Γ-images of $\boldsymbol{x}$ is preserved due to continuity of Γ, a topographic organisation also emerges (locally) for models $p(\cdot|\boldsymbol{x})$. Function Γ is realised as a RBF network:

$$\Gamma(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x}), \tag{6.18}$$

where matrix $\boldsymbol{W} \in \mathbb{R}^{6\times M}$ contains the free parameters of the model (6 is the number of parameters in $\{M_1, q, e, \imath, \omega, \rho\}$), and $\boldsymbol{\phi}(.) = (\phi_1(.), ..., \phi_M(.))^T, \phi_m(.) : \mathbb{R}^2 \rightarrow \mathbb{R}$ is an ordered set of M non-parametric nonlinear smooth basis functions. However, this mapping does not produce valid parameter vectors $\Gamma(\boldsymbol{x})$ that belong to parameter space $\mathcal{M}$, since the output of the RBF network is unbounded. Therefore, we introduce a vector-valued version of the sigmoid function g:

$$g(\boldsymbol{y}) = \left[\frac{1}{1+\exp(-y_1)},\ \frac{1}{1+\exp(-y_2)},\ \ldots, \frac{1}{1+\exp(-y_Y)}\right]^T, \tag{6.19}$$

that takes the output of the RBF network and "squashes" it between $[0,1]^Y$ where Y is the number of parameters of input $\boldsymbol{y}$. Here the input $\boldsymbol{y}$ to g is the output of the RBF network. Next, vector $g(\boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x}))$ of "squashed" values needs to be scaled to the range of the parameters. This is done by multiplying it with a diagonal matrix that has as diagonal elements the range of each parameter:

$$\boldsymbol{A} = diag[(100-0.5),\ (1-0),\ (1-0),\ (2\pi-0),\ (\frac{\pi}{2}-0),\ (100-0.5)]$$

where the ranges of the model parameters are listed in the order of primary mass, mass ratio, eccentricity, argument, inclination and period. Finally, we still need to translate the resulting values in order to obtain valid values by adding a vector $\boldsymbol{v}$ that contains the minimum value of each parameter (in the same order aforementioned):

$$\boldsymbol{v} = [0.5,\ 0.0,\ 0.0,\ 0.0,\ 0.0,\ 0.5]^T.$$

Taking into consideration these transformations, we redefine mapping Γ:

$$\Gamma(\boldsymbol{x}) = \boldsymbol{A}g(\boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x})) + \boldsymbol{v}, \tag{6.20}$$

that takes a latent point $\boldsymbol{x}$ in space $\mathcal{V}$ to a valid parameter vector $\Gamma(\boldsymbol{x})$ that addresses a noise model in $\mathcal{M}$. Thus, $\boldsymbol{\Theta}$ has become a function of the weight matrix $\boldsymbol{W}$ of the RBF network,

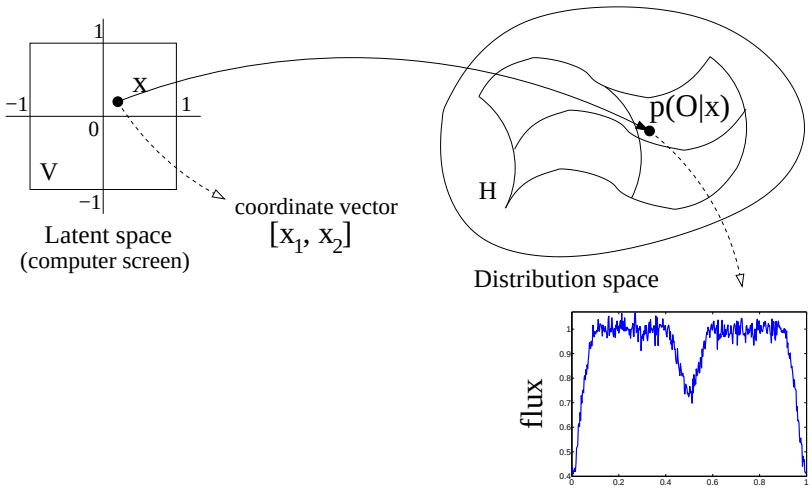


Fig. 6.4: Formulation of the topographic mapping model.

$\boldsymbol{\Theta}(\boldsymbol{W})$. Thus, (6.16) now reads:

$$p(\boldsymbol{\Theta}(\boldsymbol{W})|\mathcal{D}) \quad \propto \quad \sum_{n=1}^{N} \log \sum_{c=1}^{C} p(\boldsymbol{O}^{(n)}|\boldsymbol{x}_c) + \log p(\boldsymbol{\Theta}(\boldsymbol{W})), \tag{6.21}$$

where the likelihood is now a function of $\boldsymbol{W}$ rather than $\boldsymbol{\Theta}$.

Fig. 6.4 summarises the model formulation. Each point $\boldsymbol{x}$ of latent space $\mathcal{V}$ is non-linearly and smoothly mapped via Γ to model parameters that identify noise model $p(\cdot|\boldsymbol{x})$. The induced parameters are constrained on a two-dimensional manifold $\mathcal{M}$ that is a subspace of $\mathcal{H}$, the space of all possible parametrisations of our model. Thus, latent space $\mathcal{V}$ is embedded via Γ in $\mathcal{H}$ as a two-dimensional manifold. We refer to this GTM extension as GTM-flux.

6.3 Training of the Model

MAP estimation requires the maximisation of (6.21). This can be achieved by adopting an EM formulation of the problem by writing the (complete data) likelihood in terms of hidden indicator variables z:

$$z_c^n = \begin{cases} 1, & \text{if light curve } \boldsymbol{O}^{(n)} \text{ was generated by model } c; \\ 0, & \text{otherwise.} \end{cases}$$

We refer to the variables z collectively as $\mathcal{Z}$ which is considered missing data (missing in the sense that we have no information on the model-origin of the data points). $\mathcal{Z}$ in conjunction with training set $\mathcal{D}$ form the complete data. Using (6.13), (6.15) the (scaled) complete data

likelihood $\mathcal{L}(\boldsymbol{\Theta}(\boldsymbol{W})|\mathcal{D},\mathcal{Z})$ or in simpler notation $\mathcal{L}(\boldsymbol{W}|\mathcal{D},\mathcal{Z})$ and its logarithm read:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{W}|\mathcal{D},\mathcal{Z}) &\propto \prod_{n=1}^{N}\prod_{c=1}^{C} p(\boldsymbol{O}^{(n)}|\boldsymbol{x}_c)^{z_c^n} \\
&\propto \prod_{n=1}^{N}\prod_{c=1}^{C}\prod_{t=1}^{T} \mathcal{N}(\boldsymbol{O}^{(n)}(t); f_{\Gamma(\boldsymbol{x}_c)}(t), \sigma^2)^{z_c^n}, \\
\log \mathcal{L}(\boldsymbol{W}|\mathcal{D},\mathcal{Z}) &\propto \sum_{n=1}^{N}\sum_{c=1}^{C}\sum_{t=1}^{T} z_c^n \log \mathcal{N}(\boldsymbol{O}^{(n)}(t); f_{\Gamma(\boldsymbol{x}_c)}(t), \sigma^2).
\end{aligned} \tag{6.22}$$

As a reminder, we note that the presence of a prior term does not affect the E-step and modifies only the M-step [Ng et al., 2004]. In the E-step, the hidden variables are estimated by their posterior expectation given the observed data $\mathcal{D}$ and parameters $\boldsymbol{W}^{(i)}$ at the i-th iteration:

$$E[z_c^n|\boldsymbol{W}^{(i)}] = p(\boldsymbol{x}_c|\boldsymbol{O}^{(n)}, \boldsymbol{W}^{(i)}). \tag{6.23}$$

The posterior of latent point $\boldsymbol{x}_c$ given observation $\boldsymbol{O}^{(n)}$ is estimated in (6.23). Taking the expectation with respect to the posterior distribution of hidden variables $\mathcal{Z}$ given the current parameters $\boldsymbol{W}^{(i)}$ and observed data $\mathcal{D}$, we express the (scaled) expected complete-data log-likelihood of the model as:

$$E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{W}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{W}^{(i)}] \propto \sum_{n=1}^{N}\sum_{c=1}^{C}\sum_{t=1}^{T} p(\boldsymbol{x}_c|\boldsymbol{O}^{(n)}, \boldsymbol{W}^{(i)}) \log \mathcal{N}(\boldsymbol{O}^{(n)}(t); f_{\Gamma(\boldsymbol{x}_c)}(t), \sigma^2). \tag{6.24}$$

In the M-step we maximise:

$$E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{W}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{W}^{(i)}] + \log p(\boldsymbol{\Theta}(\boldsymbol{W})), \tag{6.25}$$

Normally, in the M-step, the derivatives of (6.25) are calculated with respect to the RBF network parameters $\boldsymbol{W}$. However, in this case, the M-step cannot be carried out analytically due to the nature of the physical model formulation in section 6.1. It is noted in [Ng et al., 2004] that the EM algorithm does not necessarily require that an optimum be achieved in the M-step; it is sufficient that the likelihood is merely improved. For our purposes we resort to numerical optimisation by employing an evolutionary algorithm.

Having trained the model, it can be used for visualisation by calculating the posterior prob-

ability of each latent point $\boldsymbol{x}$ given an observation $\boldsymbol{O}^{(n)}$ using Bayes' theorem:

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{O}^{(n)}) &= \frac{P(\boldsymbol{x})p(\boldsymbol{O}^{(n)}|\boldsymbol{x})}{p(\boldsymbol{O}^{(n)})} = \frac{P(\boldsymbol{x})p(\boldsymbol{O}^{(n)}|\boldsymbol{x})}{\sum_{c'=1}^{C} P(\boldsymbol{x}_{c'})p(\boldsymbol{O}^{(n)}|\boldsymbol{x}_{c'})} \\ &= \frac{p(\boldsymbol{O}^{(n)}|\boldsymbol{x})}{\sum_{c'=1}^{C} p(\boldsymbol{O}^{(n)}|\boldsymbol{x}_{c'})}. \end{aligned} \tag{6.26}$$

Each data item $\boldsymbol{O}$ is represented in the latent space $\mathcal{V}$ by a point $proj(\boldsymbol{O})$ given by the expectation of the posterior distribution over all points $\boldsymbol{x}_c, c = 1, \ldots, C$ of the latent grid in $\mathcal{V}$:

$$proj(\boldsymbol{O}) = \sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{O})\boldsymbol{x}_c. \tag{6.27}$$

6.4 Experimentation

6.4.1 Datasets

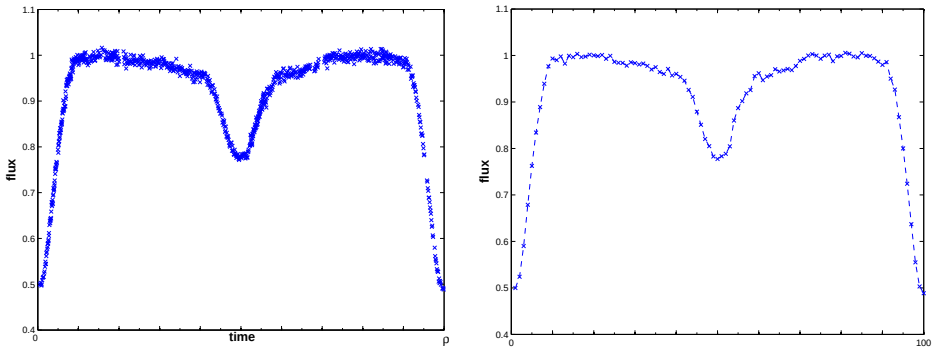
We performed experiments on two datasets. Dataset *1* is a synthetic, toy dataset that consists of 200 light curves. A common set of model parameters, $\{M_1 = 5, q = 0.8, e = 0.3, \imath = \frac{\pi}{2}\}$ was defined. However, two distinct values $\rho_1 = 2, \rho_2 = 5$ of period and $\omega_1 = 0, \omega_2 = \frac{5}{6}\pi$ of argument of periastron were used, to create 4 classes of light curves (50 in each class) by the combinations of these values, $\{\rho_1, \rho_2\} \times \{\omega_1, \omega_2\}$. The discerning characteristic of each class is the position of each secondary eclipse and the widths of the eclipses. Each light curve was then generated by these sets of parameters after adding to them Gaussian noise in order to introduce some variability. Gaussian noise with standard deviation of 0.075 was also subsequently added to the light curves themselves to simulate observational errors.

Dataset *2* consists of 71 light curves from real observations obtained from two resources available[3] on the WWW, the *Catalogue and Archive of Eclipsing Binaries* at http://ebola.eastern.edu/ and the *All Sky Automated Survey* at http://archive.princeton.edu/ asas/.

6.4.2 Preprocessing

Dataset *2* of observed light curves was preprocessed before training. Preprocessing is necessary as the light curves in dataset *2* are afflicted with observational errors such as noise, gaps and overlapping of flux points. Such observational errors are not accounted for by our model and must be treated before the data are to be used. To this purpose we employed linear interpolation as a simple way of remedying these problems. Fig. 6.5 displays the outcome of linear interpolation on

[3]Last accessed on the 12th September 2007.



(a) (b)

Fig. 6.5: Raw, noisy light curve from dataset that has been phase-shifted so that its first point is the point of the primary eclipse (a). Performing linear interpolation on the raw light curve yields the light curve in (b).

a real light curve of dataset *2*. After linear interpolation, light curves must also be phase-shifted so that their first point is the primary eclipse and resampled to equal length as described in section 6.1.3.

Light curves in dataset *2* were resampled at 100 regular intervals which was judged an adequate sample rate. While increasing the sampling rate retains more detail of the raw dataset, it must be noted that it also takes a toll on training time. Furthermore, training becomes more prone to numerical underflow when calculating likelihoods as the number of samples increases. This is due to the multiplication of many probabilities in the calculation of the likelihoods (see (6.13)) necessary for the posterior probabilities $p(\boldsymbol{x}_c|\boldsymbol{O}^{(n)}, \boldsymbol{W}^{(i)})$ (see (6.23)). This can be especially problematic in the initial iterations of training because the probabilities involved in the likelihood calculation can be very low, when the model poorly fits the data. Thus, to avoid this numerical problem in the implementation a minimum must be imposed on the probabilities e.g. $\min(\epsilon, p(\boldsymbol{x}_c|\boldsymbol{O}^{(n)}, \boldsymbol{W}^{(i)}))$, where we set $\epsilon = 10^{-10}$.

6.4.3 Initialisation

A good initialisation strategy is important for the EM algorithm. We found that the following initialisation protocol significantly aided training:

1. *Dataset fitting*: For each light curve $\boldsymbol{O}^{(n)}$, $n = 1, 2, ..., N$ in the dataset we fit regression model $f_{\boldsymbol{\theta}}(t)$ using an evolutionary algorithm [Rowe and Hidović, 2004] and obtain a parameter vector $\boldsymbol{\theta}^{(n)}$. Likelihood $p(\boldsymbol{O}|\boldsymbol{\theta})$ is the objective function maximised during the fitting. The resulting parameter vectors are collected in the set $\Upsilon = \{\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, ..., \boldsymbol{\theta}^{(N)}\}$.

2. *SOM training*: We employ SOM [Kohonen, 1990] to organise the parameter vectors Υ obtained at step 1. The topology of SOM is defined to be identical to the grid of latent points in $\mathcal{V}$, thus there are C neurons on the SOM lattice that share the same indexing $c = 1, \ldots, C$ as latent points $\boldsymbol{x}_c$. Having trained SOM on Υ we obtain topographically organised codebook vectors $\boldsymbol{B}_c$ from neurons $c = 1, \ldots, C$.

3. *Weight initialisation*: We require that Γ maps each latent point $\boldsymbol{x}_c$ to the codebook vector corresponding to neuron c. To that purpose we optimise weight matrix $\boldsymbol{W}$ accordingly. We denote the $i-$th row of $\boldsymbol{W}$ by $\boldsymbol{W}_i$. We note that in (6.20) each parameter θ_i of the parameter vector $\boldsymbol{\theta}$ depends only on $\boldsymbol{W}_i$. Thus, we can optimise $\boldsymbol{W}$ in a row-wise manner. The error for parameter θ_i when optimising $\boldsymbol{W}_i$ is the mean squared error:

$$MSE(\theta_i) = \sum_{c=1}^{C} (\Gamma_i(\boldsymbol{x}_c) - \boldsymbol{B}_{c,i})^2, \tag{6.28}$$

   where $\Gamma_i(\boldsymbol{x}_c)$ and $\boldsymbol{B}_{c,i}$ indicate the $i-$th component of $\Gamma(\boldsymbol{x}_c)$ and $\boldsymbol{B}_c$ respectively. The error is minimised by scaled conjugate gradients. In order to avoid overly large initial weights in matrix $\boldsymbol{W}$ (and hence a potentially complex initial map Γ), we constrain the optimisation of all elements of $\boldsymbol{W}$ in $[-5, 5]$.

Even though the topographic organisation in step 2 uses an inappropriate Euclidean notion of distance for organising the parameter vectors from step 1 (two parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ may be close in the Euclidean sense, but the models that they instantiate may lead to different fluxes), this constitutes a useful heuristic to roughly approximate the range of parameters of the local models (step 1), as well as their spatial relation (step 2).

6.4.4 Training

The lattice was a 10×10 regular grid (i.e. $C = 100$) and the RBF network consisted of $M = 17$ basis functions; 16 of them were Gaussian radial basis functions of variance $\sigma^2 = 1$ centred on a 4×4 regular grid, and one was a bias term. Parameters were initialised using the initialisation method described in section 6.4.3. In the M-step we did not used a gradient-based optimisation procedure as we did for GTM-HMTM and GTM-MTM. Instead we employed an evolutionary algorithm described in [Rowe and Hidović, 2004]. The fitness function was given by (6.25).

The evolutionary algorithm in [Rowe and Hidović, 2004] uses a single member population and a real number representation for the search space. What is special about this algorithm is the probability distribution that it uses for generating an offspring candidate from the single parent.

This particular probability distribution is motivated, roughly speaking, by the following property of Gray codes in genetic algorithms that use binary string representations: flipping a single bit produces mostly small incremental changes, but occastionally larger jumps in the search space may be made. The probability distribution emulates this property (amongst others) which helps it escape local minima in the search space. The algorithm possess a single free parameter called precision. This parameter corresponds to the string length in the binary representation: the greater the precision, the longer the binary string and therefore the finer the exploration of the search space and the closer to the optimum we can get. However, too great a precision can be inefficient in reaching the optimum, as in the binary string length analogy this means that a large number of mutations is necessary to reach the optimum. Therefore, setting the precision is a delicate task that balances on one hand the quality of solution and on the other hand computational time.

The evolutionary algorithm in [Rowe and Hidović, 2004] is particularly suited for optimisation problems where the parameters are bounded. Since in our problem we do not have any bounds, we tried to confine the parameters in ranges $[-u, +u]$ for $u \in \{1, 5, 10, 20, 50\}$. We also tried different values for precision between 100 and 1200. We found that in practice the best values for range and precision were $[-10, +10]$ and 400 respectively. Moreover, we set the number of iterations of the evolutionary algorithm to 1000.

The E-step complexity of GTM-flux is $\mathcal{O}(CNT)$ which follows from (6.13) and the number C of latent points and number N of data items in the dataset. In practice, the training requirements of a MATLAB (version 7.3) implementation of GTM-flux were in hours 1 for the toy dataset *1* and 12 for dataset *2* when run on a machine equipped with an Athlon XP 3000+ CPU and 512MB of memory.

6.4.5 Results

Fig. 6.6 is the topographic map constructed for the toy dataset. Each point stand for a light curve projected to latent space $\mathcal{V}$ and is coloured according to class membership. Also next to each cluster, a typical light curve has been plotted. The classes have been identified and organised appropriately, each occupying one of the four corners of the plot. Starting clockwise from the top left corner, we first come across the class corresponding to parameters $\{\rho_2, \omega_1\}$, then $\{\rho_1, \omega_1\}$, then $\{\rho_1, \omega_2\}$ and finally $\{\rho_2, \omega_2\}$. In Fig. 6.7 we see six plots, each corresponding to one of the six parameters of the light curve model. Each plot is a heat[4] map and illustrates how its corresponding parameter varies over the topographic map. As mentioned in 6.4.1, the

[4]The rank of colours ranges from white to yellow to red to black corresponding from high to low values.

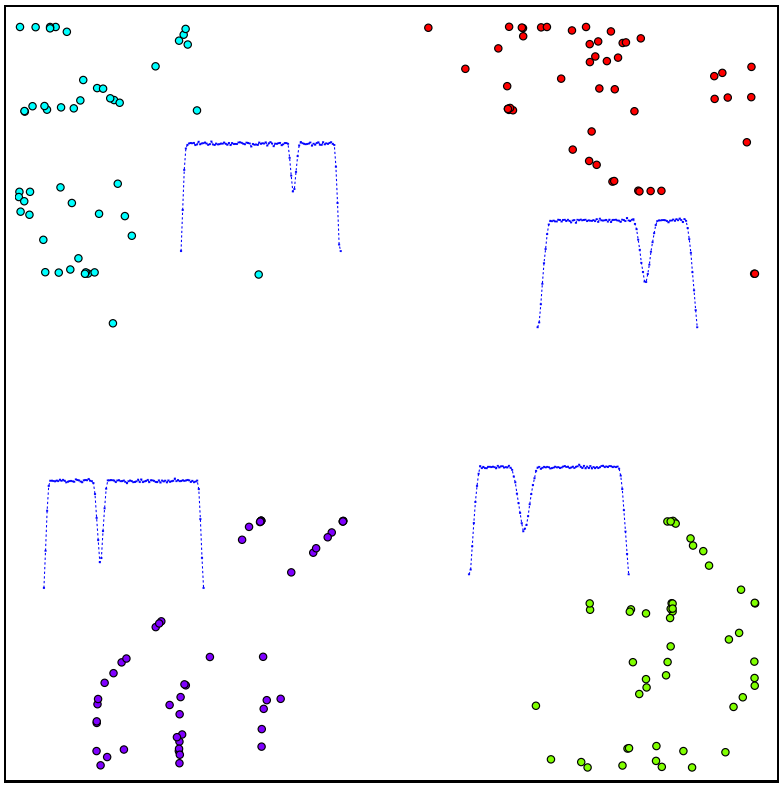


Fig. 6.6: Visualisation of toy dataset. A representative light curve is plotted next to each cluster.

toy dataset is composed of four classes by combining two distinct values for period and argument of periastron. The separation of the classes based on these characteristics, is depicted in the period and argument of periastron plots in Fig. 6.7. Furthermore, in Fig. 6.8, the light curves of the underlying local noise models $p(\cdot|\boldsymbol{x})$ are depicted, fully conforming to the visualisation of Fig. 6.6.

Fig. 6.9 displays the visualisation of the dataset of real light curves, with the projected real light curves in red and the light curves corresponding to the underlying local noise models in black. We choose this type of plot instead of that in Fig. 6.6, since real light curves exhibit more variability than the toy data, and thus it is more informative to see them drawn on the visualisation plot. Here, the projections in Fig. 6.9 of the real light curves do not exhibit as clear an organisation as in the case of the toy data. Nevertheless, we note certain trends. At the top left corner of Fig. 6.9 we note the presence of light curves with both relatively shallow primary and secondary eclipses. In Fig. 6.10 the plots of primary mass and mass ratio show that the respected area constitutes of models of low masses that naturally leads to stars of low luminosity and flat-looking light curves. Moving from there towards the right of the plot, we find a few light curves with a more well-defined, deeper eclipse, and upon reaching the middle we

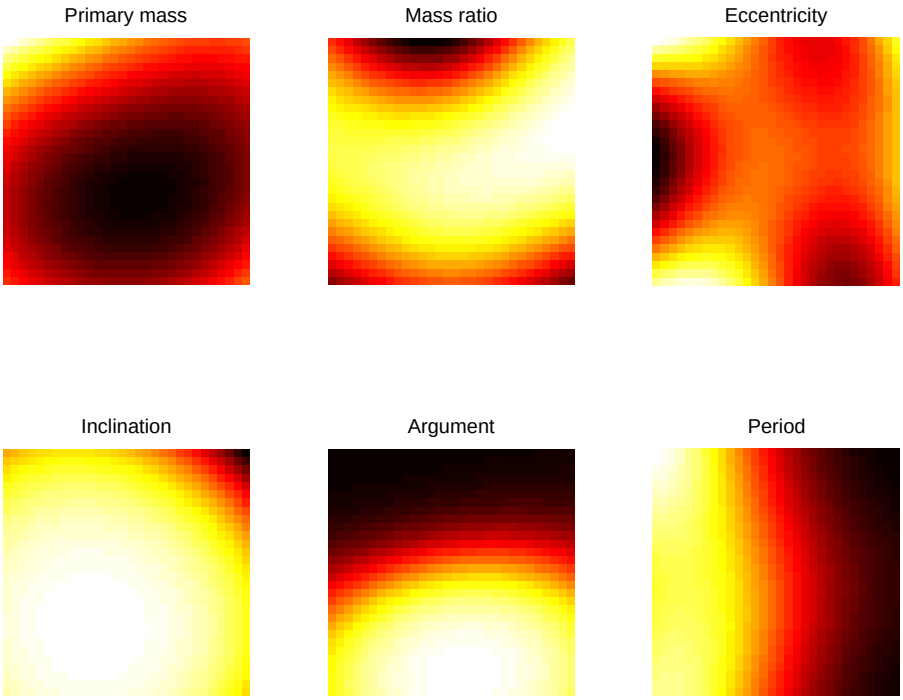


Fig. 6.7: Toy dataset: Heat maps of model parameters. Hot and cold regions signify high and low values respectively. We note how the plots of argument and period reflect the organisation of the map in Fig. 6.6.

find light curves with both primary and secondary eclipses almost equally deep. Moving further to the far right of the plot, the primary eclipses are gaining in depth as primary mass increases and the secondary eclipses become shallower as the mass ratio decreases, again in accordance to the respective plots in Fig. 6.10. On the other hand, moving from the top middle towards the centre, the depths of both eclipses are almost equal, in agreement with the high mass ratio in Fig. 6.10. However, the eclipses are narrower compared to the wider eclipses of the light curves encountered in the upper right corner. This is explained by the decreasing primary mass of the underlying local noise models as shown in Fig. 6.10. The lower half of the graph, as indicated by the light curves of the underlying local noise models in black (where the secondary eclipse does not occur at centre of light curve) in Fig. 6.9 and the eccentricity plot in Fig. 6.10, is dominated by light curves that exhibit eccentric orbits. However, at the lower right part of the plot there is a point where a concentration of light curves occurs that have narrow and almost equally deep primary and secondary eclipses and exhibit little or no eccentricity. The reason that they are projected there seems to be the high argument values seen in Fig. 6.10 which "cancel out" the effect of eccentricity by shifting the secondary eclipse close to the middle of the light curve. This is exhibited by the light curves in black corresponding to the underlying local noise models of this location in Fig. 6.9. Upon inspection of these particular light curves projected there,

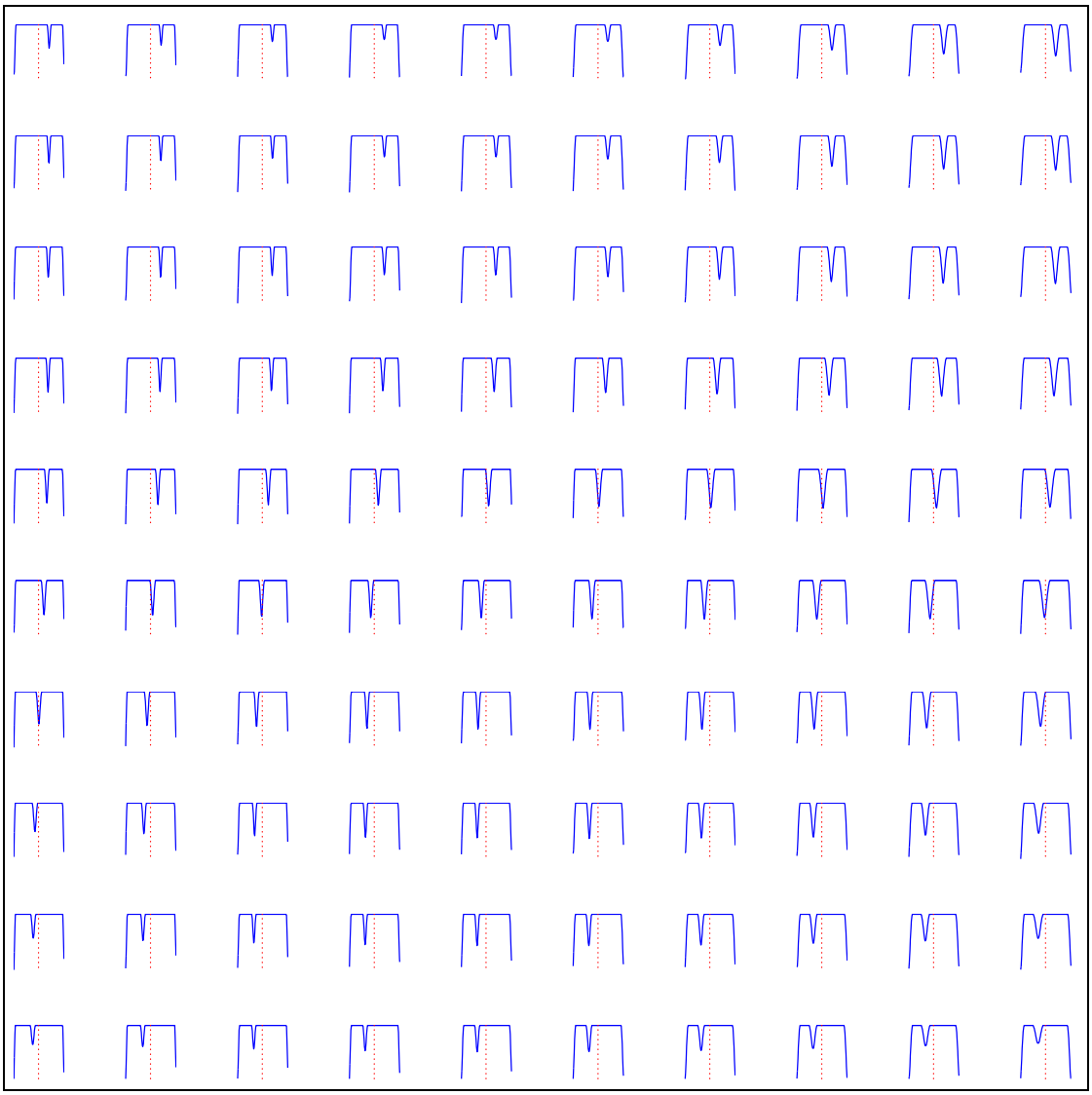


Fig. 6.8: Toy dataset: Topographic map of underlying noise models.

their high periods seems to be the reason why they are projected in these regions instead of somewhere closer to their similar looking counterparts at the centre of the visualisation plot. High periods are favoured in this region as confirmed by the period plot in Fig. 6.10. The few light curves projected above and right of this concentration point are light curves of very high periods. At the lower left part of the plot, close to the centre, we come across a few eccentric light curves and at the very far left we find some light curves that are arranged almost like on a column. By inspecting the black curves of the underlying local noise models in Fig. 6.9, our expectation is to find eccentric light curves in this region. However, not all of these light curves are eccentric. The two light curves closest to the bottom-left, have an almost flat secondary eclipse due to very low mass ratio and its difficult to understand why the are projected there

Fig. 6.9: Visualisation of dataset *2* of real data. Light curves in red are the projected real data and light curves in black are the light curves of the underlying local noise models.

(although their low mass ratio is understandable). The two light curves that are right above seem to fit the eccentric profiles of the light curves of the underlying local noise models and are appropriately positioned there. However, the light curves at the top of this column close to the left edge of the plot, appear incongruous. These light curves have very high periods and light curves of such periods require a finer sampling in order to capture their eclipses in detail. Otherwise, for any sparser sampling only very few points correspond to the eclipses making the processing and subsequent projection (after training) of such light curves problematic. This appears to be the case for these light curves.

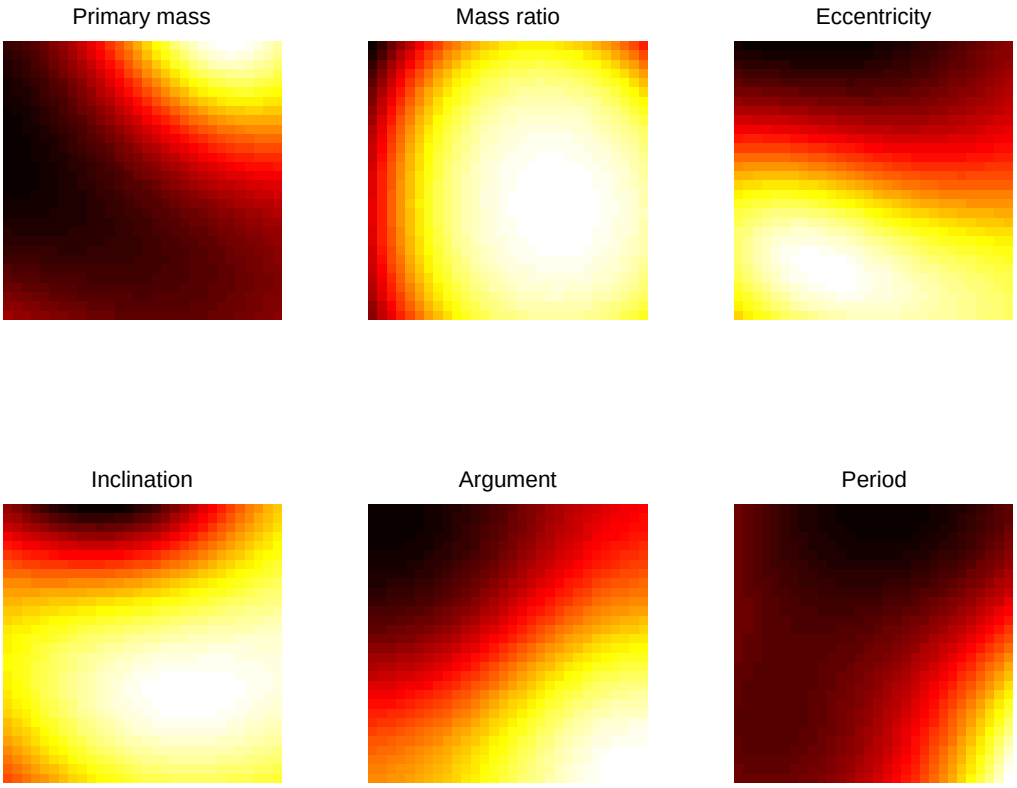


Fig. 6.10: Real dataset: Heat maps of model parameters. Hot and cold regions signify high and low values respectively.

6.5 Magnification Factors

Similarly to GTM and its extensions in chapter 5, we calculate magnification factors following both approaches of Fisher information matrix (FIM) and Kullback-Leibler divergence (KLD).

6.5.1 Fisher Information

Each latent point has two coordinates $\boldsymbol{x} = [x_1, x_2]$ that are mapped via a smooth non-linear mapping Γ to a noise model $p(\cdot|\boldsymbol{x})$ on manifold $\mathcal{M}$ of all noise models. As discussed in section 5.3, FIM is defined as the matrix:

$$\begin{pmatrix} \frac{\partial^2 \log p(\cdot|\boldsymbol{x})}{\partial x_1^2} & \frac{\partial^2 \log p(\cdot|\boldsymbol{x})}{\partial x_1 \partial x_2} \\ \frac{\partial^2 \log p(\cdot|\boldsymbol{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 \log p(\cdot|\boldsymbol{x})}{\partial x_2^2} \end{pmatrix}$$

Due to the nature of the physical model, closed-form derivatives are unattainable. Thus, we resort to the numerical calculation of the second-order partial derivatives using difference

approximations [Burden, 1997]:

$$\frac{\partial^2}{\partial x_r^2}\log p(\cdot|\boldsymbol{x}) = \frac{\log p(\cdot|\boldsymbol{x}+[h,\ 0]^T) - \log p(\cdot|\boldsymbol{x}-[h,\ 0]^T)}{2h}, \tag{6.29}$$

and

$$\begin{aligned}\frac{\partial^2}{\partial x_r \partial x_s}\log p(\cdot|\boldsymbol{x}) &= \frac{\log p(\cdot|\boldsymbol{x}+[h,\ h]^T) - \log p(\cdot|\boldsymbol{x}+[h,\ -h]^T)}{4h^2} \\ &- \frac{\log p(\cdot|\boldsymbol{x}+[-h,\ h]^T) + \log p(\cdot|\boldsymbol{x}+[-h,\ -h]^T)}{4h^2},\end{aligned} \tag{6.30}$$

where $r, s \in \{1, 2\}$ with $r \neq s$ and h is a sufficiently small positive constant, e.g. $h = 10^{-6}$.

In order to illustrate the magnification factors on manifold $\mathcal{M}$, we followed the perturbation scheme of chapter 5, illustrated in Fig. 5.2. Here we perturb each latent centre $\boldsymbol{x}_c$ in 16 regularly spaced directions on a small circle (we have set its radius to 10^{-5}).

6.5.2 Kullback-Leibler Divergence

As stated in section 5.2.1, for two multivariate Gaussian distributions $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathcal{N}(\cdot; \boldsymbol{\mu}', \boldsymbol{\Sigma}')$, KLD can be calculated as:

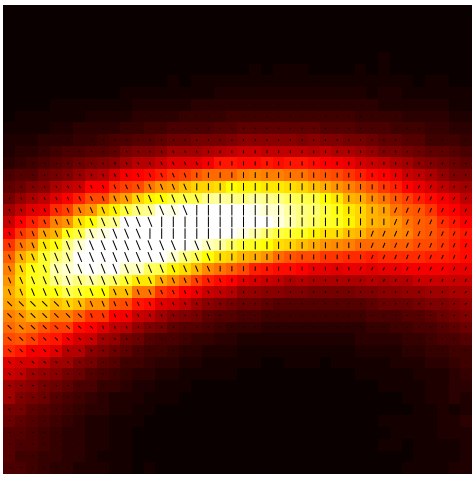
$$D_{KL}[\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})||\mathcal{N}(\cdot; \boldsymbol{\mu}', \boldsymbol{\Sigma}')] = \frac{1}{2}\big[\log(\frac{det\boldsymbol{\Sigma}'}{det\boldsymbol{\Sigma}}) - d + tr(\boldsymbol{\Sigma}'^{-1}\boldsymbol{\Sigma}) + (\boldsymbol{\mu}-\boldsymbol{\mu}')^T\boldsymbol{\Sigma}'^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}')\big].$$

Since noise model $p(\boldsymbol{O}|\boldsymbol{x})$ is a homoscedastic multivariate Gaussian distribution, the KLD between two noise models, $p(\boldsymbol{O}|\boldsymbol{x})$ and $p(\boldsymbol{O}|\boldsymbol{x}+\boldsymbol{dx})$ simplifies to the inner product of the corresponding means:

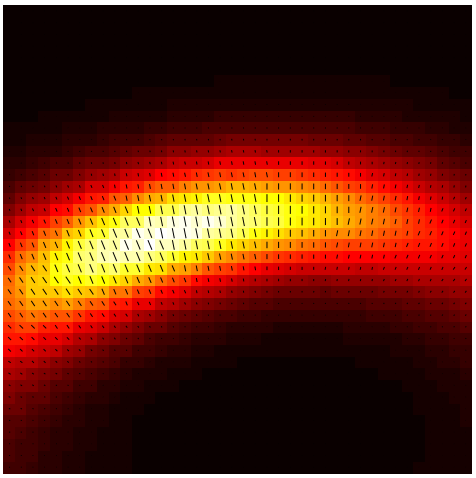
$$\begin{aligned}D_{KL}[p(\boldsymbol{O}|\boldsymbol{x})||p(\boldsymbol{O}|\boldsymbol{x}+\boldsymbol{dx})] &= \frac{1}{2\sigma}(f_{\Gamma(\boldsymbol{x})} - f_{\Gamma(\boldsymbol{x}+\boldsymbol{dx})})^T(f_{\Gamma(\boldsymbol{x})} - f_{\Gamma(\boldsymbol{x}+\boldsymbol{dx})}) \\ &\propto (f_{\Gamma(\boldsymbol{x})} - f_{\Gamma(\boldsymbol{x}+\boldsymbol{dx})})^T(f_{\Gamma(\boldsymbol{x})} - f_{\Gamma(\boldsymbol{x}+\boldsymbol{dx})}),\end{aligned} \tag{6.31}$$

where $f_{\Gamma(\boldsymbol{x})}$ is the light curve produced by noise model $p(\boldsymbol{O}|\boldsymbol{x})$ (addressed by latent point $\boldsymbol{x}$) and $f_{\Gamma(\boldsymbol{x}+\boldsymbol{dx})}$ is the light curve produced by noise model $p(\boldsymbol{O}|\boldsymbol{x}+\boldsymbol{dx})$ (addressed by latent point $\boldsymbol{x}+\boldsymbol{dx}$).

The same procedure as for the Fisher information matrix is applied for GTM-flux, namely perturbing a latent point in 16 regularly spaced directions on a small circle (again the radius is 10^{-5}) and measuring the KLD between the original noise model addressed by $\boldsymbol{x}$ and the perturbed model addressed by $\boldsymbol{x}+\boldsymbol{dx}$.



(a)



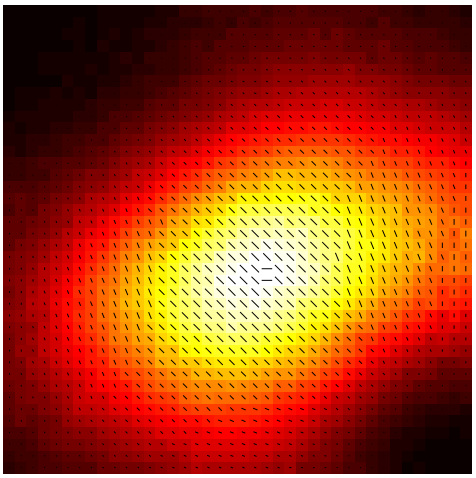
(b)

Fig. 6.11: Magnification factors as heat maps for toy dataset based on Fisher information (a) and KLD approximation (b).

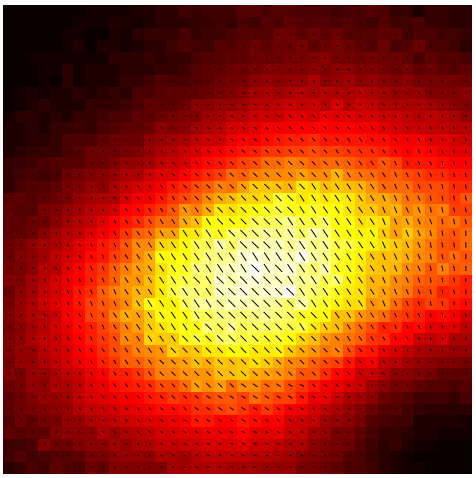
6.5.3 Results

We follow the same procedure as described in 5.7 for the demonstration of magnification factors for GTM-flux. However, we found that the calculation of FIM and KLD for GTM-flux was not as time consuming as in the previous extensions of GTM, and we decided to calculate magnification factors on a finer rectangular grid of 40×40 points.

In Fig. 6.11(a) and 6.11(b) we see the magnification factors calculated for the toy dataset based on Fisher information and KLD respectively. The two plots convey practically the same information, they both express a strong separation between the upper and lower regions. As one can see in Fig. 6.6, the upper and lower regions correspond to light curves of arguments of



(a)



(b)

Fig. 6.12: Magnification factors as heat maps for real dataset based on Fisher information (a) and KLD approximation (b).

periastron ω_1 and ω_2 respectively. Classes that differ in the argument of periastron are more distant to each other since their secondary eclipse occur at different times. On the other hand, classes that differ in period should experience a milder separation between them, since they only differ at the width of the eclipses. Therefore, the magnification factors reveal that Fig. 6.6 experiences a high vertical stretch along the middle.

In Fig. 6.12(a) and 6.12(b) we see the magnification factors calculated for the dataset of real lightcurves, based on FIM and KLD respectively. Again, the two plots are almost identical. However, they are not as informative as in the case of the toy dataset. Here we see that a strong magnification appears at the centre of the plot, gradually losing its power as we move away from the centre. No information seems to be conveyed regarding the presence of clusters. Nevertheless,

this seems to be related to the parameter plots in Fig. 6.10 where almost all parameters exhibit high variations at the centres of their respective plots. Thus, the magnifications in Fig. 6.12 effectively summarise this volatile behaviour of all parameters of the local noise models at the centre of the map.

6.6 Discussion

In [Brett et al., 2004], topographic maps of eclipsing binary light curves are constructed using SOM. The light curves are treated as vectorial data and SOM is applied in its standard form, i.e. weights are updated by the usual update-rule, neighbourhood is a Gaussian kernel while the learning rate and neighbourhood radius are monotonically decreased. The only difference lies in the determination of the winning neuron. When a light curve is presented to SOM, the squared Euclidean distance is calculated between the light curve and the weight vector of each neuron. After this comparison, the same light curve is phase-shifted by one sample point and the squared Euclidean distances are calculated again. This procedure is iterated until the light curve has been shifted by all increments (which eventually returns the original light curve). The winning neuron is then the neuron that minimises the squared Euclidean distance between its weight and any of the phase-shifted versions of the light curve. This alteration in determining the winning neuron is necessary as the light curves in the dataset are not preprocessed and standardised to some reference phase. This application of SOM organises light curves according to shape. However, it does not convey any physical interpretation of how the light curves relate and could in fact be used to organise any kind of curves since no specific domain knowledge on eclipsing binary stars in incorporated. Also, inspecting the codebooks of the formed map indicates only what kind of shapes are expected to be mapped in the affinity of the corresponding neurons, but no information on model parameters is provided. Moreover, it is very likely that the training of SOM leads to codebooks that are not valid light curves, which makes the interpretation of the map more difficult. For example, a codebook that is close to two light curves with different shapes and is updated to increase its future response when they are presented again, could form an invalid light curve as illustrated in Fig. 6.13.

On the other hand, GTM-flux is a model-based approach. It employs a special local noise model which drives the topographic organisation of the map. The underlying "codebooks" (i.e. local noise models) correspond to templates of physical models. A clear notion of metric is defined on the topographic map, based on the distances between the probabilistic local noise models. This facilitates a principled interpretation of the visualisation plots Fig. 6.6 and 6.9 which is further supplemented by the parameter plots in Fig. 6.7 and 6.10 and the light curve

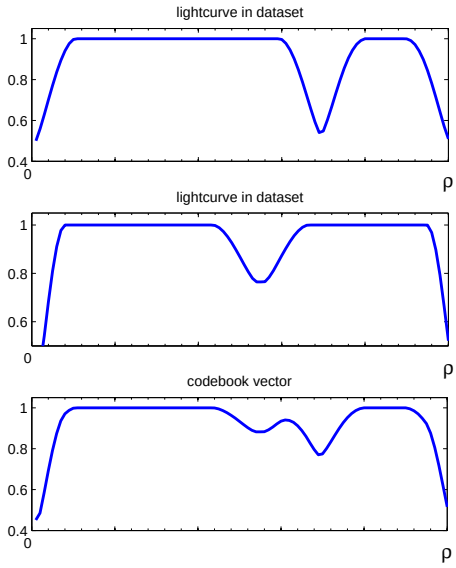


Fig. 6.13: SOM for light curves: codebooks may form invalid light curves.

plot in Fig. 6.8. Moreover, magnification factors can be calculated, illustrated in Fig. 6.11 and 6.12, that reveal contractions/expansions in the latent space. Furthermore GTM-flux offers control to the user over the metric induced by the local noise models, and could incorporate other physical aspects of eclipsing binary systems such as non-spherical stars. Moreover, similar to GTM-HMTM and GTM-MTM, GTM-flux has also the capability of detecting overfitting (even though it was not demonstrated for the GTM-flux).

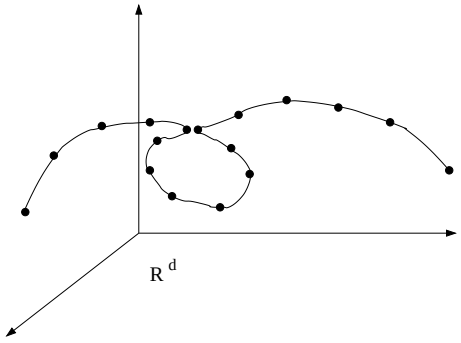


Fig. 6.14: A one-dimensional latent space embedded in a higher-dimensional latent space. Two fairly distant points of the latent space are mapped to similar locations in the high-dimensional space which can cause a defect to the topographic map.

There is a limitation in GTM-flux that we would like to mention. There appears to be a problem of identifiability in the local noise models. That is, noise models of different parameters can generate very similar light curves. This makes the formation of a map problematic as distant regions of the map may contain noise models that are statistically (almost) identical

and therefore should model the density of the same data points. In such cases, defects may be manifested in the topographic map, such as the map folding in order to bring distant regions close together. This is demonstrated in Fig. 6.14 in the case of one-dimensional latent space embedded in a high-dimensional data space. This problem is a potential explanation for the not completely successful topographic organisation of the real light curves in section 6.4.5. Moreover, this problem is not restricted in the case of GTM-flux, but may also be manifested in any GTM formulation where the local noise models are not identifiable.

Chapter 7

Conclusions

The GTM algorithm sets a paradigm for the development of topographic maps based on generative probabilistic model-based formulations. Following this, we developed two methods for topographic organisation of tree-structured data, the GTM-HMTM and the GTM-MTM. We also studied a real world problem and developed a model, the GTM-flux, for the topographic organisation of astronomical lightcurves from eclipsing binary stars. Compared with the recursive neural-based approaches discussed in chapter 2, there are a number of advantages when following a generative probabilistic model-based approach.

In the GTM paradigm a form of local noise model must be chosen that bears a certain plausibility in terms of generation of the dataset. The local noise models in the original GTM are Gaussian densities that generate vectorial data. In GTM-HMM [Tiňo et al., 2004] where topographic maps of sequences are constructed, HMMs are chosen. In chapter 4, we have employed HMTMs and MTMs as candidate generative processes for tree-structured data. Alternative local noise model formulations allow the user to express in a principled way a notion of structured data similarity that will be driving the topographical organisation in visualisation plots. Thus, for the successful construction of a topographic map, a prerequisite is the selection of a suitable generative probabilistic model. For instance, if one was studying a corpus of sentences from various sources of origin (e.g. dialects) and wanted to construct a topographic map of such data items, one possible choice for local noise models would be probabilistic context free grammars [Manning and Schtze, 1999]. In chapter 6 we demonstrated this by formulating a probabilistic physical model for the construction of topographic maps of astronomical light curves.

The choice of local noise model imparts a form of control over the shaping of visualisation plots. In the GTM paradigm this is done by imposing the form of local noise models. For GTM-HMTM two data items are viewed as "similar", if they are highly probable under the

same local HMTM. Of course, it is possible to have different notions of similarity even for the same data set. It can well be that there are two users that would like to see the same data items organised on the visualisation plot using different criteria for item similarity, depending on what aspects of the data they are interested in. Then it is up to the user to formulate the appropriate noise model and let the data visualisation be driven by it. It is difficult to quantify what the induced notion of similarity is in recursive neural-based approaches. For instance in recursive-neural based approaches such as SOMSD, what is the distance metric that places some trees close together and some further apart? Can one have some form of control over the shaping of visualisation plots in such approaches?

The model-based nature of GTMs allows great deal of transparency of the visualisation plot formation and a principled interpretation of the data visualisations. A number of useful plots are readily provided regarding the learnt structure of the underlying local noise models, such as plots of state-transitions and means of emissions as illustrated in section 4.3.3, that aid the interpretation of the map. Recursive model formulations along the lines of SOMSD do not lend themselves naturally to obtaining insights of this sort.

Moreover, in our approach the optimisation of the free parameters of GTM-HMTM/MTM/flux is driven by a well defined cost function, which is the negative log of the model likelihood. Training may be performed using a wide range of optimisation algorithms. In chapter 4 we trained GTM-HMTM and GTM-HMT using gradient methods while in chapter 6 we trained GTM-flux using gradient-free methods. We also note that an advantage of probabilistic model formulation is the possibility to inspect the tendency of the model to overfit the training data, by measuring the log-likelihood on an independent validation set. Despite the high number of parameters in the GTM formulations (for GTM-HMTM there are $M(K + K^2 + Kd)$ number of parameters), we found that in practice this does not present a significant difficulty in training the model because of its highly constrained nature. What seems to make training difficult is the lack of a good initialisation procedure.

Having trained a GTM model via EM, the data points are projected from the data space on the latent space. To this end we calculate the responsibilities (posterior probabilities) of the underlying local noise models. Our mixture of noise models is a constrained mixture, constrained by the smooth two-dimensional structure of the latent space. Hence, neighbouring latent points correspond to noise models that lead to similar answers (responsibilities) when queried about a certain data point. Each data point can then be placed at the mean location of the responsibilities in order to reflect the contribution of all local models. The same also applies to a newly incoming data point. Clearly, this is a transparent method in the sense that we can understand

why a certain point was placed in a particular position in the visualisation (latent) space by inspecting the underlying local models. This level of transparency is not readily provided when visualising new trees using SOMSD.

The generative nature of our approach allows further data exploration after a first impression of the visualisation through hierarchical visualisation in the spirit of [Tiño and Nabney, 2002]. Also, the incorporation of priors on the model parameters is straightforward. MAP estimation is possible by adding to the expected complete-data log-likelihood an additional term, namely the log-likelihood of the prior density on the parameters.

Furthermore, the fact that the non-linear mapping of the GTM formulation from the latent space to the local model space is smooth allows us to calculate magnification factors (chapter 5). We have presented two approaches toward this end, precise Fisher information matrix and KLD approximation, which have been verified by experiments. This constitutes a useful tool for the study of clusters and can be used to further interpret the visualisation plot as magnifications of the manifold of local models are not detectable from the visualisation plots alone. The presence of low magnification in a certain region can help us infer the presence of a potential cluster as we expect the generative process of the underlying local models to change slightly as we move in that region. Thus, we expect neighbouring models to model roughly the same distribution of the partition of data present in that certain area. On the contrary, high magnification signifies the volatility of the local models and hence that data points in the region are expected to differ significantly from each other.

The GTM extensions that we formulated in this work, GTM-HMTM, GTM-MTM and GTM-flux suffer from certain limitations. One such limitation is the long time required to train them which was observed during the experiments that we performed. As we saw in section 4.2, GTM-HMTM is a double-hidden model; there is one set of hidden variables for the GTM part of the model, and a set of hidden variables for the HMTM local noise model. This introduces a considerable computational burden to GTM-HMTM during the E-step. On the other hand in section 4.5, GTM-MTM was computationally less demanding than GTM-HMTM because MTMs do not have hidden states as HMTMs do. In GTM-flux, section 6.2, the local noise model does not have hidden underlying process either, but training was costly as it was carried out with an evolutionary optimisation method.

In connection to this, another limitation of the presented GTM extensions is their lack of a principled initialisation method. A good initialisation would offer a headstart to the training algorithm, limiting training time and also eliminating the need of re-running the algorithm several times with different initial random weights. A certain remedy to this, was proposed in

the case of GTM-flux. Of course, this method could be transferred and applied to GTM-HMTM and GTM-MTM as well. As aforementioned in section 6.4.3, we fit a noise model to each data item, retrieve the sets of parameters of all fitted models and use SOM to produce a map of topographically organised model parameters. This is purely a heuristic method, with an incorrect notion of similarity is used, namely the Euclidean distance between model parameters, in order to introduce topographic organisation in the model space. Nevertheless, in the experiments of GTM-flux we found that it did provide a much better start than mere random initialisation. GTM on the other hand, has an initialisation procedure, which initialises the weights so that the GTM approximates principal component analysis [Bishop et al., 1998].

Another limitation, briefly mentioned in section 6.6, concerns the identifiability of the local noise models. By identifiability we refer to the notion that if two probability density models $p(\cdot|\theta)$ and $p(\cdot|\theta')$ are equal then it must follow that $\theta = \theta'$, i.e. the parameter vector uniquely identifies a probability model. For example, mixtures of Gaussians are not identifiable in the sense that reordering the components results in a different parameter vector, even though the mixture still models the same density as before. This also applies to HMMs, HMTMs and MTMs where states may be reordered without any consequences to the modelled density. This problem of identifiability may be manifested in the GTM extensions that employ these models as local noise models, as a topological defect in the constructed map. For two latent points $\boldsymbol{x}$ and $\boldsymbol{x}'$ that reside in different locations in latent space $\mathcal{V}$, it may occur that their respective local noise models are (almost) identical $p(\cdot|\boldsymbol{x}) \approx p(\cdot|\boldsymbol{x}')$. In this case the map sustains a topological defect in the sense that two remote locations are (almost) connected. An approximate way of detecting such occurrences is by defining a regular rectangular grid of points in the latent space and measuring the KLD distance between all pairs of points. Of course the finer the grid is, the better the approximation will be. Thus, if for two latent points $\boldsymbol{x}$ and $\boldsymbol{x}'$ that belong to different neighbourhoods we measure $D_{KL}[p(\cdot|\boldsymbol{x})||p(\cdot|\boldsymbol{x}')] \approx 0$, then this signifies a topographic defect in the map. A potential solution to this problem involves imposing constraints on the parameters so that a certain ordering of model components/parameters is enforced that ensures identifiability.

Apart from addressing the above limitations, there also other issues open to investigation for the future. One evident direction is of course the accommodation of additional data types such as graphs or the creation of custom data types for specialised applications. However, apart from the practical interest and perhaps the potential challenges in implementing efficiently the processing of the new data types, the main framework would in essence be the same as in the GTM extensions presented in this thesis. In actual fact, we have already seen that the GTM

and its extensions (GTM-HMTM, GTM-MTM, GTM-HMM and GTM-flux) all share the same basic components: the latent space $\mathcal{V}$, a smooth mapping Γ, calculation of likelihoods and the EM algorithm. Accommodating a new data type involves defining a new appropriate mapping Γ that induces the parameters of the local noise models at hand, employing noise model dependent algorithm for the calculation of likelihoods and posteriors and employing a suitable optimisation algorithm for the M-step.

A more interesting line of research that could deal with domains where long training times are required (such as the GTM-HMTM in section 4.2), is the adoption of a hierarchical visualisation scheme. Given such a domain, one could train a top level GTM with local noise models of limiting modelling capability but that would allow fast calculation of the model likelihoods for the input data. Although such a map would not reflect accurately enough the topographic organisation of the data, it could quickly impart a first impression. Further refinement of the top level map could then be facilitated in the spirit of [Tiño and Nabney, 2002]. After the user has indicated regions of interest on the map that he/she would like to examine in greater detail, a second level of children GTMs could be built. However, this time the new GTMs would not employ the same local noise models as the top level ones, but more refined noise models that would lead to more detailed topographic maps. This refinement could be repeated in this fashion and allow the user to explore certain regions of the maps in increasing detail. In conjunction with this scheme, if a suitable initialisation strategy could be devised to initialise the parameters of the GTMs in a more informative way than mere random initialisation, the computational time savings would be significant.

A further enhancement to investigate is the incorporation of class information that may be available on a subset of the dataset. The idea is that data items that belong to the same class should be located (relatively) close to each other on the map as membership to the same class must indicate a certain affinity between them. This kind of information should be taken into account to guide the learning of the topographic maps. Tentatively speaking, such information could be employed in the E-step so that neighbouring latent points express similar posterior probabilities over data items of the same class.

Finally, another interesting direction would be the investigation of alternative map configurations. In GTM and its presented extensions the latent space $\mathcal{V}$ is the square $[-1, +1]$ that is mapped via a smooth mapping Γ a two-dimensional manifold of parameters. However, other kinds of manifolds are admittable that elude the scope of the present work. One possibility is the consideration of data that lie on manifolds with one or multiple "holes" [1] (that do not

[1] Roughly speaking this is known as the *genus* number in topology.

disconnect the manifold) or even data that lie on multiple manifolds.

We conclude by summarising the main benefits of the generative probabilistic model-based approach followed in this work, over recursive neural-based approaches:

- Trained models are topographic maps endowed with a well defined distance metric.
- There is a well defined cost function driving the model training that can be used for principled model comparison.
- Trained models can be checked in a natural way for possible overfitting by comparing log-likelihoods on training and validation sets.
- The smooth mapping from the latent to the data space enables the calculation of magnification factors, a useful tool that supplements our understanding of the visualisation plots.
- The methodology can be extended to include hierarchical visualisations for detailed user-guided exploration of subsets of data.

Appendix A

Derivatives for Constrained Mixture Example

Regarding the first example of a mixture of Gaussians constrained on a straight line in section 3.5: We define $Q(\mathbf{\Theta}, \mathbf{\Theta}^{(i)}) = E_{\mathcal{Z}}[\log \mathcal{L}(\mathbf{\Theta}|\mathcal{D}, \mathcal{Z})|\mathcal{D}, \mathbf{\Theta}^{(i)}]$. The derivative of Q with respect to x_c, $c = 1, \ldots, C$ is:

$$
\begin{aligned}
&\frac{\partial}{\partial x_c} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(i)}) = 0, \\
&\frac{\partial}{\partial x_c} \sum_{n=1}^{N} \sum_{c'=1}^{C} p(c'|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \log p(\boldsymbol{t}^{(n)}|c') = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{\partial}{\partial x_c} \log p(\boldsymbol{t}^{(n)}|c) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{\partial}{\partial x_c} \log \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma}) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{1}{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma})} \frac{\partial}{\partial x_c} \mathcal{N}(\boldsymbol{t}^{(n)}; l(x_c), \boldsymbol{\Sigma}) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{1}{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma})} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma}) \frac{\partial}{\partial x_c} (-\frac{1}{2} (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))^T \boldsymbol{\Sigma}^{-1} \frac{\partial}{\partial x_c} (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c)) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))^T \boldsymbol{\Sigma}^{-1} \begin{bmatrix} 1 \\ \alpha \end{bmatrix} = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \begin{bmatrix} t_1^{(n)} - x_c \\ t_2^{(n)} - \alpha x_c + \beta\alpha \end{bmatrix}^T \begin{bmatrix} \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_3 \end{bmatrix} \begin{bmatrix} 1 \\ \alpha \end{bmatrix} = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) [(t_1^{(n)} - x_c)(\sigma_1 + \alpha\sigma_2) + (t_2^{(n)} - \alpha x_c - \beta)(\sigma_2 + \alpha\sigma_3)] = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) [(t_1^{(n)} - x_c)(\sigma_1 + \alpha\sigma_2) + \alpha(t_1^{(n)} - x_c)(\sigma_2 + \alpha\sigma_3)] = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (t_1^{(n)} - x_c)(\sigma_1 + 2\alpha\sigma_2 + \alpha^2\sigma_3) = 0, \\
&\underbrace{(\sigma_1 + 2\alpha\sigma_2 + \alpha^2\sigma_3)}_{>0} \sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (t_1^{(n)} - x_c) = 0, \\
&\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (t_1^{n} - x_c) = 0, \\
&x_c = \frac{\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) t_1^{(n)}}{\sum_{n=1}^{N} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})}.
\end{aligned} \tag{A.1}
$$

where $t_1^{(n)}$ is the first coordinate of $\boldsymbol{t}^{(n)} = [t_1^{(n)},\ t_2^{(n)}]^T$. The quantity noted with the underbrace is a positive constant. This can be seen by considering that since covariance matrix $\boldsymbol{\Sigma}$ is a

positive definite matrix and consequently its inverse is too, we know that:

$$\begin{aligned}
\begin{bmatrix}1\\ \alpha\end{bmatrix}^T \boldsymbol{\Sigma}^{-1} \begin{bmatrix}1\\ \alpha\end{bmatrix} &> 0,\\
\begin{bmatrix}1\\ \alpha\end{bmatrix}^T \begin{bmatrix}\sigma_1 & \sigma_2\\ \sigma_2 & \sigma_3\end{bmatrix} \begin{bmatrix}1\\ \alpha\end{bmatrix} &> 0,\\
(\sigma_1 + 2\alpha\sigma_2 + \alpha^2\sigma_3) &> 0.
\end{aligned} \tag{A.2}$$

The derivative of Q with respect to α is:

$$\begin{aligned}
&\frac{\partial}{\partial\alpha} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(i)}) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{\partial}{\partial\alpha} \log p(\boldsymbol{t}^{(n)}|c) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{\partial}{\partial\alpha} \log \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma}) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{1}{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma})} \frac{\partial}{\partial\alpha} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma}) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) \frac{1}{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma})} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma}) \frac{\partial}{\partial\alpha} (-\frac{1}{2}(\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (-(\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c)))^T \boldsymbol{\Sigma}^{-1} \frac{\partial}{\partial\alpha} (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c)) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))^T \boldsymbol{\Sigma}^{-1} \begin{bmatrix}0\\ x_c\end{bmatrix} = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))^T \begin{bmatrix}\sigma_1 & \sigma_2\\ \sigma_2 & \sigma_3\end{bmatrix} \begin{bmatrix}0\\ x_c\end{bmatrix} = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) x_c[(t_1^{(n)} - x_c)\sigma_2 + (t_2^{(n)} - \alpha x_c - \beta)\sigma_3] = 0,\\
&\sigma_2 \sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) x_c (t_1^{(n)} - x_c) + \sigma_3 \sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)}) (t_2^{(n)} - \alpha x_c - \beta) x_c = 0.
\end{aligned} \tag{A.3}$$

The first summand is equal to zero after updating each x_c according to (A.1). For the second

summand we have:

$$\alpha = \frac{\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})(t_2^{(n)} - \beta)x_c}{\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})x_c^2}. \tag{A.4}$$

The derivative of Q with respect to β is:

$$\begin{aligned}
&\frac{\partial}{\partial\beta}Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(i)}) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})\frac{\partial}{\partial\beta}\log p(\boldsymbol{t}^{(n)}|c) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})\frac{\partial}{\partial\beta}\log \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma}) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})\frac{1}{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma})}\frac{\partial}{\partial\beta}\mathcal{N}(\boldsymbol{t}^{(n)}; l(x_c), \boldsymbol{\Sigma}) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})\frac{1}{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma})}\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{l}(x_c), \boldsymbol{\Sigma})\frac{\partial}{\partial\beta}(-\frac{1}{2}(\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})(-(\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c)))^T\boldsymbol{\Sigma}^{-1}\frac{\partial}{\partial\beta}(\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c)) = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})(\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))^T\boldsymbol{\Sigma}^{-1}\begin{bmatrix}0\\1\end{bmatrix} = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})(\boldsymbol{t}^{(n)} - \boldsymbol{l}(x_c))^T\begin{bmatrix}\sigma_1 & \sigma_2\\ \sigma_2 & \sigma_3\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = 0,\\
&\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})[(t_1^{(n)} - x_c)\sigma_2 + (t_2^{(n)} - \alpha x_c - \beta)\sigma_3] = 0,\\
&\sigma_2\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})(t_1^{(n)} - x_c) + \sigma_3\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})(t_2^{(n)} - \alpha x_c - \beta) = 0.
\end{aligned} \tag{A.5}$$

The first summand is equal to zero by setting each x_c according to (A.1). For the second summand we have:

$$\beta = \frac{\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})(t_2^{(n)} - \alpha x_c)}{\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\boldsymbol{t}^{(n)}, \boldsymbol{\Theta}^{(i)})} \tag{A.6}$$

Appendix B

Derivatives for GTM-HMTM

Here we give a detailed derivation of the derivatives encountered in section 4.2.2 at the M-step for the GTM-HMTM:

▷ Element $w_{j,m}$ in matrix $\boldsymbol{W}^{(\boldsymbol{\pi})}$:

$$
\begin{aligned}
&\frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}]}{\partial w_{j,m}} \\
= &\frac{\partial}{\partial w_{j,m}} \sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \Bigg[\sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \log p(q_1=k|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \\
+ &\sum_{u=2}^{U_n}\sum_{k=1}^{K}\sum_{l=1}^{K} p(q_u=l,q_{\rho(u)}=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \log p(q_u=l|q_{\rho(u)}=k,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \\
+ &\sum_{u=1}^{U_n}\sum_{k=1}^{K} p(q_u=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \log p(\boldsymbol{o}_u^{(n)}|q_u=k,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \Bigg] \\
= &\sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \frac{\partial}{\partial w_{j,m}} \log p(q_1=k|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \\
= &\sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \frac{1}{p(q_1=k|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})} \frac{\partial}{\partial w_{j,m}} p(q_1=k|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}).
\end{aligned}
\tag{B.1}
$$

Calculate derivative $\frac{\partial}{\partial w_{j,m}} p(q_1=k|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})$:

$$
\begin{aligned}
&\frac{\partial}{\partial w_{j,m}} p(q_1 = k|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) = \frac{\exp(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))}{\sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))} \\
=& \frac{\left(\frac{\partial}{\partial w_{j,m}} \exp(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\right) \sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c)) - \exp(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\left(\frac{\partial}{\partial w_{j,m}} \sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)}{\left(\sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)^2} \\
=& \frac{\delta_{j,k}\phi_m(\boldsymbol{x}_c)\exp(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c)) \sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c)) - \exp(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\phi_m(\boldsymbol{x}_c)\exp(\boldsymbol{W}_j^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))}{\left(\sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)^2} \\
=& \frac{\delta_{j,k}\phi_m(\boldsymbol{x}_c)\exp(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c)) \sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))}{\left(\sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)\left(\sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)} \\
-& \frac{\exp(\boldsymbol{W}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\phi_m(\boldsymbol{x}_c)\exp(\boldsymbol{W}_j^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))}{\left(\sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)\left(\sum_{l=1}^K \exp(\boldsymbol{W}_l^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)} \\
=& \delta_{j,k}\phi_m(\boldsymbol{x}_c)p(q_1 = k|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) - \phi_m(\boldsymbol{x}_c)p(q_1 = k|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}})p(q_1 = j|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \\
=& \phi_m(\boldsymbol{x}_c)p(q_1 = k|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}})(\delta_{j,k} - p(q_1 = j|\boldsymbol{x}_c, \boldsymbol{\mathcal{W}})). \qquad \text{(B.2)}
\end{aligned}
$$

Substitute (B.2) in (B.1):

$$
\begin{aligned}
&\frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}]}{\partial w_{j,m}} \\
= &\sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \frac{1}{p(q_1=k|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})} \\
\times &\ \phi_m(\boldsymbol{x}_c) p(q_1=k|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})(\delta_{j,k} - p(q_1=j|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})) \\
= &\sum_{n=1}^{N}\sum_{c=1}^{C} \phi_m(\boldsymbol{x}_c) p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})(\delta_{j,k} - p(q_1=j|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})) \\
= &\sum_{n=1}^{N}\sum_{c=1}^{C} \phi_m(\boldsymbol{x}_c) p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \\
\times &\left(\sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})\delta_{j,k} - \sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) p(q_1=j|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \right) \\
= &\sum_{n=1}^{N}\sum_{c=1}^{C} \phi_m(\boldsymbol{x}_c) p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \\
\times &\left(p(q_1=j|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) - p(q_1=j|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \cancelto{1}{\sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})} \right) \\
= &\sum_{n=1}^{N}\sum_{c=1}^{C} \phi_m(\boldsymbol{x}_c) p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \left(p(q_1=j|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) - p(q_1=j|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \right), \qquad \text{(B.3)}
\end{aligned}
$$

where we have used the fact that $\sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) = 1$.

▷ Element $w_{j,m}^{(r)}$ in matrix $\boldsymbol{W}^{(\boldsymbol{B}_r)}$:

$$
\begin{aligned}
&\frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}]}{\partial w_{j,m}^{(r)}} \\
= &\frac{\partial}{\partial w_{j,m}^{(r)}} \sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \Bigg[\sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \log p(q_1=k|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \\
+ &\sum_{u=2}^{U_n}\sum_{k=1}^{K}\sum_{l=1}^{K} p(q_u=l,q_{\rho(u)}=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \log p(q_u=l|q_{\rho(u)}=k,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \\
+ &\sum_{u=1}^{U_n}\sum_{k=1}^{K} p(q_u=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \log p(\boldsymbol{o}_u^{(n)}|q_u=k,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \Bigg] \\
= &\sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \\
\times &\sum_{u=2}^{U_n}\sum_{l=1}^{K} p(q_u=l,q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \frac{\partial}{\partial w_{j,m}^{(r)}} \log p(q_u=l|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \\
= &\sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \\
\times &\sum_{u=2}^{U_n}\sum_{l=1}^{K} p(q_u=l,q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \frac{1}{p(q_u=l|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})} \frac{\partial}{\partial w_{j,m}^{(r)}} p(q_u=l|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}).
\end{aligned}
\tag{B.4}
$$

Calculate derivative $\frac{\partial}{\partial w_{j,m}^{(r)}} p(q_u=l|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})$:

$$
\begin{aligned}
&\frac{\partial}{\partial w_{j,m}^{(r)}} p(q_u = l | q_{\rho(u)} = r, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) = \frac{\exp(\boldsymbol{W}_l^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c))}{\sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c))} \\
= & \frac{\left(\frac{\partial}{\partial w_{j,m}^{(r)}} \exp(\boldsymbol{W}_l^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right) \sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c))}{\left(\sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right)^2} \\
- & \frac{\exp(\boldsymbol{W}_l^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \left(\frac{\partial}{\partial w_{j,m}^{(r)}} \sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right)}{\left(\sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right)^2} \\
= & \frac{\delta_{l,j} \exp(\boldsymbol{W}_l^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \phi_m(\boldsymbol{x}_c) \sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c))}{\left(\sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right)^2} \\
- & \frac{\exp(\boldsymbol{W}_l^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \exp(\boldsymbol{W}_j^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \phi_m(\boldsymbol{x}_c)}{\left(\sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right)^2} \\
= & \frac{\delta_{l,j} \exp(\boldsymbol{W}_l^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \phi_m(\boldsymbol{x}_c) \sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c))}{\left(\sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right) \left(\sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right)} \\
- & \frac{\exp(\boldsymbol{W}_l^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \exp(\boldsymbol{W}_j^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \phi_m(\boldsymbol{x}_c)}{\left(\sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right) \left(\sum_{k=1}^K \exp(\boldsymbol{W}_k^{(\boldsymbol{B}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right)} \\
= & \delta_{l,j} p(q_u = l | q_{\rho(u)} = r, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \phi_m(\boldsymbol{x}_c) - p(q_u = l | q_{\rho(u)} = r, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) p(q_u = j | q_{\rho(u)} = r, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \phi_m(\boldsymbol{x}_c) \\
= & \phi_m(\boldsymbol{x}_c) p(q_u = l | q_{\rho(u)} = r, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) (\delta_{l,j} - p(q_u = j | q_{\rho(u)} = r, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}})). \qquad \text{(B.5)}
\end{aligned}
$$

Substitute (B.5) in (B.4):

$$
\begin{aligned}
&\frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}]}{\partial w_{j,m}^{(r)}} \\
&= \sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \sum_{u=2}^{U_n}\sum_{l=1}^{K} p(q_u=l, q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \\
&\times \frac{1}{p(q_u=l|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})}\phi_m(\boldsymbol{x}_c)p(q_u=l|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})(\delta_{l,j}-p(q_u=j|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})) \\
=&\sum_{n=1}^{N}\sum_{c=1}^{C} \phi_m(\boldsymbol{x}_c)p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \sum_{u=2}^{U_n}\sum_{l=1}^{K} p(q_u=l, q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})(\delta_{l,j}-p(q_u=j|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})) \\
&= \sum_{n=1}^{N}\sum_{c=1}^{C} \phi_m(\boldsymbol{x}_c)p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \\
&\times \sum_{u=2}^{U_n}\Bigg(\sum_{l=1}^{K} p(q_u=l, q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})\delta_{l,j} \\
&- \sum_{l=1}^{K} p(q_u=l, q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})p(q_u=j|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})\Bigg) \\
&= \sum_{n=1}^{N}\sum_{c=1}^{C} \phi_m(\boldsymbol{x}_c)p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \\
\times&\sum_{u=2}^{U_n}\Bigg(p(q_u=j, q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) - p(q_u=j|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})\sum_{l=1}^{K} p(q_u=l, q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})\Bigg) \\
&= \sum_{n=1}^{N}\sum_{c=1}^{C} \phi_m(\boldsymbol{x}_c)p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \\
&\times \sum_{u=2}^{U_n}\Bigg(p(q_u=j, q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) - p(q_u=j|q_{\rho(u)}=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})p(q_{\rho(u)}=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})\Bigg),
\end{aligned}
\tag{B.6}
$$

where we have used the fact that $\sum_{l=1}^{K} p(q_u = l, q_{\rho(u)} = r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) = p(q_{\rho(u)} = r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})$.

▷ Element $w_{j,m}^{(r)}$ in matrix $\boldsymbol{W}^{(\boldsymbol{\psi}_r)}$:

$$
\begin{aligned}
&\frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}]}{\partial w_{j,m}^{(r)}} \\
= \;& \frac{\partial}{\partial w_{j,m}^{(r)}} \sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \Bigg[\sum_{k=1}^{K} p(q_1=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \log p(q_1=k|\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \\
+ \;& \sum_{u=2}^{U_n}\sum_{k=1}^{K}\sum_{l=1}^{K} p(q_u=l,q_{\rho(u)}=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \log p(q_u=l|q_{\rho(u)}=k,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \\
+ \;& \sum_{u=1}^{U_n}\sum_{k=1}^{K} p(q_u=k|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \log p(\boldsymbol{o}_u^{(n)}|q_u=k,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \Bigg] \\
= \;& \sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \sum_{u=1}^{U_n} p(q_u=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \frac{\partial}{\partial w_{j,m}^{(r)}} \log p(\boldsymbol{o}_u^{(n)}|q_u=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}) \\
= \;& \sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)}) \\
\times \;& \sum_{u=1}^{U_n} p(q_u=r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)}) \frac{1}{p(\boldsymbol{o}_u^{(n)}|q_u=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})} \frac{\partial}{\partial w_{j,m}^{(r)}} p(\boldsymbol{o}_u^{(n)}|q_u=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}). \qquad \text{(B.7)}
\end{aligned}
$$

Calculate derivative $\frac{\partial}{\partial w_{j,m}^{(r)}} p(\boldsymbol{o}_u^{(n)}|q_u=r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})$:

$$
\begin{aligned}
& \frac{\partial}{\partial w_{j,m}^{(r)}} p(\boldsymbol{o}_u^{(n)} | q_u = r, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \\
= & \frac{\partial}{\partial w_{j,m}^{(r)}} \mathcal{N}(\boldsymbol{o}_u^{(n)}; \boldsymbol{\mu}_r(\boldsymbol{x}_c), \boldsymbol{\Sigma}_r(\boldsymbol{x}_c)) = \frac{\partial}{\partial w_{j,m}^{(r)}} \mathcal{N}(\boldsymbol{o}_u^{(n)}; \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c), \boldsymbol{\Sigma}_r(\boldsymbol{x}_c)) \\
= & \frac{\partial}{\partial w_{j,m}^{(r)}} \left(\frac{1}{2\pi} \right)^{D/2} \|\boldsymbol{\Sigma}\|^{-\frac{1}{2}} \exp \left(-\frac{1}{2} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right) \\
= & \left(\frac{1}{2\pi} \right)^{D/2} \|\boldsymbol{\Sigma}\|^{-\frac{1}{2}} \exp \left(-\frac{1}{2} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right) \\
\times & \frac{\partial}{\partial w_{j,m}^{(r)}} \left(-\frac{1}{2} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right) \\
= & \mathcal{N}(\boldsymbol{o}_u^{(n)}; \boldsymbol{\mu}_r(\boldsymbol{x}_c), \boldsymbol{\Sigma}_r(\boldsymbol{x}_c)) \frac{\partial}{\partial w_{j,m}^{(r)}} \left(-\frac{1}{2} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right) \\
= & p(\boldsymbol{o}_u^{(n)} | q_u = r, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \frac{\partial}{\partial w_{j,m}^{(r)}} \left(-\frac{1}{2} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right) \quad \text{(B.8)} \\
= & p(\boldsymbol{o}_u^{(n)} | q_u = r, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \left(\cancel{-\frac{1}{2}} 2 (\boldsymbol{E}^{jm} \boldsymbol{\phi}(\boldsymbol{x}_c))^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)) \right) \quad \text{(B.9)} \\
= & p(\boldsymbol{o}_u^{(n)} | q_u = r, \boldsymbol{x}_c, \boldsymbol{\mathcal{W}}) \phi_m(\boldsymbol{x}_c) (\boldsymbol{E}_j^{jm})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)} \boldsymbol{\phi}(\boldsymbol{x}_c)). \quad \text{(B.10)}
\end{aligned}
$$

In line (B.8) we make use of the chain rule, and two formulas of derivatives of vectors and matrices. These two formulas, which can be found in [Petersen and Pedersen, 2006], are:

- the derivative of $\frac{\partial \boldsymbol{a}^T \boldsymbol{B} \boldsymbol{a}}{\partial \boldsymbol{a}}$, where $\boldsymbol{a}$ is a vector and $\boldsymbol{B}$ a matrix, is

$$
\frac{\partial \boldsymbol{a}^T \boldsymbol{B} \boldsymbol{a}}{\partial \boldsymbol{a}} = (\boldsymbol{B} + \boldsymbol{B}^T) \boldsymbol{a}. \quad \text{(B.11)}
$$

  If $\boldsymbol{B}$ is a symmetric, then:

$$
\frac{\partial \boldsymbol{a}^T \boldsymbol{B} \boldsymbol{a}}{\partial \boldsymbol{a}} = 2 \boldsymbol{B} \boldsymbol{a}. \quad \text{(B.12)}
$$

  This is the case in line (B.8), since covariance matriced are inherently symmetric.

- the derivative of a matrix $\boldsymbol{B}$ with respect to one of its elements b_{ij} is:

$$
\frac{\partial \boldsymbol{B}}{\partial b_{ij}} = \boldsymbol{E}^{ij}, \quad \text{(B.13)}
$$

  where $\boldsymbol{E}^{ij}$ is a matrix of zeros, apart from element (i, j) that is equal to 1.

We apply the chain rule to $\frac{\partial}{\partial w_{j,m}^{(r)}}\Big(-\frac{1}{2}(\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c))^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c))\Big)$ of line (B.8). Let $\boldsymbol{a} = (\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c))$. Take derivative according to (B.12):

$$\frac{\partial}{\partial \mathrm{a}_i}\Big(-\frac{1}{2}\boldsymbol{a}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{a}\Big) = \Big(\frac{\partial \boldsymbol{a}^T}{\partial \mathrm{a}_i}\boldsymbol{\Sigma}^{-1}\boldsymbol{a}\Big). \tag{B.14}$$

Substitute for $\boldsymbol{a}$ and use (B.13):

$$\begin{aligned}
&\frac{\partial}{\partial w_{j,m}^{(r)}}\Big(-\frac{1}{2}(\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c))^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c))\Big)\\
=\ &\Big(\frac{\partial}{\partial w_{j,m}^{(r)}}(\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c))^T\Big)\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c))\\
=\ &(\boldsymbol{E}^{jm}\boldsymbol{\phi}(\boldsymbol{x}_c))^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c))\\
=\ &\boldsymbol{\phi}(\boldsymbol{x}_c)^T(\boldsymbol{E}^{jm})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c)).
\end{aligned} \tag{B.15}$$

Having explained the result in (B.10) we substitute it in (B.7):

$$\begin{aligned}
&\frac{\partial E_{\mathcal{Z}}[\log \mathcal{L}(\boldsymbol{\mathcal{W}}|\mathcal{D},\mathcal{Z})|\mathcal{D},\boldsymbol{\mathcal{W}}^{(i)}]}{\partial w_{j,m}^{(r)}}\\
=\ &\sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)})\sum_{u=1}^{U_n} p(q_u = r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})\\
\times\ &\frac{1}{p(\boldsymbol{o}_u^{(n)}|q_u = r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})}p(\boldsymbol{o}_u^{(n)}|q_u = r,\boldsymbol{x}_c,\boldsymbol{\mathcal{W}})\boldsymbol{\phi}(\boldsymbol{x}_c)^T(\boldsymbol{E}^{jm})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c))\\
=\ &\sum_{n=1}^{N}\sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)},\boldsymbol{\mathcal{W}}^{(i)})\sum_{u=1}^{U_n} p(q_u = r|\boldsymbol{y}^{(n)},\boldsymbol{x}_c,\boldsymbol{\mathcal{W}}^{(i)})\boldsymbol{\phi}(\boldsymbol{x}_c)^T(\boldsymbol{E}^{jm})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}_u^{(n)} - \boldsymbol{W}^{(\boldsymbol{\psi}_r)}\boldsymbol{\phi}(\boldsymbol{x}_c)).
\end{aligned} \tag{B.16}$$

Appendix C

Derivatives for GTM-MTM

Here we give a detailed derivation of $\frac{\partial}{\partial w_{jm}^{st}} p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}})$, encountered in section 4.5.2 at the M-step for the GTM-MTM:

$$
\begin{aligned}
& \frac{\partial}{\partial w_{jm}^{st}} p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}}) \\
= \;& \frac{\partial}{\partial w_{jm}^{st}} b_{tl}^{s} = \frac{\partial}{\partial w_{jm}^{st}} \frac{\exp(\boldsymbol{W}_{l}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))}{\sum_{l'=1}^{K}\exp(\boldsymbol{W}_{l'}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))} \\
= \;& \frac{\left(\frac{\partial}{\partial w_{jm}^{st}}\exp(\boldsymbol{W}_{l}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)\left(\sum_{l'=1}^{K}\exp(\boldsymbol{W}_{l'}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)}{\left(\sum_{l'=1}^{K}\exp(\boldsymbol{W}_{l'}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)^2} - \frac{\exp(\boldsymbol{W}_{l}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\frac{\partial}{\partial w_{jm}^{st}}\left(\sum_{l'=1}^{K}\exp(\boldsymbol{W}_{l'}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)}{\left(\sum_{l'=1}^{K}\exp(\boldsymbol{W}_{l'}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)^2} \\
= \;& \frac{\delta_{l,j}\exp(\boldsymbol{W}_{l}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\phi_m(\boldsymbol{x}_c)\left(\sum_{l'=1}^{K}\exp(\boldsymbol{W}_{l'}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)}{\left(\sum_{l'=1}^{K}\exp(\boldsymbol{W}_{l'}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)^2} - \frac{\exp(\boldsymbol{W}_{l}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\exp(\boldsymbol{W}_{j}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\phi_m(\boldsymbol{x}_c)}{\left(\sum_{l'=1}^{K}\exp(\boldsymbol{W}_{l'}^{s,t}\boldsymbol{\phi}(\boldsymbol{x}_c))\right)^2} \\
= \;& \delta_{l,j} b_{tl}^{s}\phi_m(\boldsymbol{x}_c) - b_{tl}^{s} b_{tj}^{s}\phi_m(\boldsymbol{x}_c) \\
= \;& \phi_m(\boldsymbol{x}_c)\left(\delta_{l,j} b_{tl}^{s} - b_{tl}^{s} b_{tj}^{s}\right) = \phi_m(\boldsymbol{x}_c)\left(\delta_{l,j} b_{t\mathbf{j}}^{s} - b_{tl}^{s} b_{tj}^{s}\right) \\
= \;& \phi_m(\boldsymbol{x}_c) p(\boldsymbol{o}_u = j|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s)\left(\delta_{l,j} - p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s, \boldsymbol{\mathcal{W}})\right). \qquad \text{(C.1)}
\end{aligned}
$$

Appendix D

Derivatives for Magnification Factors

The first derivative of the initial probability for state k in section 5.6.2 is:

$$\begin{aligned}
\frac{\partial}{\partial x_q} p(q_1 = k|\boldsymbol{x}) &= \frac{\partial}{\partial x_q} g_k(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x})) = \frac{\partial}{\partial x_q} \frac{\exp(\boldsymbol{A}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))}{\sum_{i=1}^K \exp(\boldsymbol{A}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))} \\
&= \frac{\exp(\boldsymbol{A}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{A}_k^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x})\sum_{i=1}^K \exp(\boldsymbol{A}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))}{\sum_{i=1}^K \exp(\boldsymbol{A}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))^2} \\
&- \frac{\exp(\boldsymbol{A}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\sum_{i=1}^K [\exp(\boldsymbol{A}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{A}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x})]}{\sum_{i=1}^K \exp(\boldsymbol{A}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))^2} \\
&= \frac{\exp(\boldsymbol{A}_k^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))}{(\sum_{i=1}^K \exp(\boldsymbol{A}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x})))} \\
&\left(\frac{\boldsymbol{A}_k^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x})\sum_{i=1}^K \exp(\boldsymbol{A}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))}{(\sum_{i=1}^K \exp(\boldsymbol{A}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x})))} - \frac{\sum_{i=1}^K [\exp(\boldsymbol{A}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{A}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x})]}{(\sum_{i=1}^K \exp(\boldsymbol{A}_i^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x})))} \right), \\
\frac{\partial}{\partial x_q} p(h_1 = k|\boldsymbol{x}) &= g_k(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))(\boldsymbol{A}_k^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^K [g_i(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{A}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_q}\boldsymbol{\phi}(\boldsymbol{x})]).
\end{aligned}$$

The second derivative for the initial probability for state k is:

$$
\begin{aligned}
\frac{\partial^2}{\partial x_q \partial x_r} p(q_1 = k|\boldsymbol{x}) &= \frac{\partial}{\partial x_q}\left\{ g_k(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\left(\boldsymbol{A}_k^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[g_i(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{A}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\right)\right\} \\
&= \left\{\frac{\partial}{\partial x_q} g_k(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\right\}\left(\boldsymbol{A}_k^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[g_i(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{A}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\right) \\
&+ g_k(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\left\{\frac{\partial}{\partial x_r}\left(\boldsymbol{A}_k^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[g_i(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{A}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\right)\right\} \\
&= \left\{\frac{\partial}{\partial x_q} g_k(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\right\}\left(\boldsymbol{A}_k^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) - \sum_{i=1}^{K}[g_i(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{A}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\right) \\
&+ g_k(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\Bigg(\boldsymbol{A}_k^{(\boldsymbol{\pi})}\frac{\partial^2}{\partial x_q \partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) \\
&- \sum_{i=1}^{K}[\frac{\partial}{\partial x_r} g_i(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{A}_i^{(\boldsymbol{\pi})}\frac{\partial}{\partial x_r}\boldsymbol{\phi}(\boldsymbol{x}) + g_i(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}))\boldsymbol{A}_i^{(\boldsymbol{\pi})}\frac{\partial^2}{\partial x_q \partial x_r}\boldsymbol{\phi}(\boldsymbol{x})]\Bigg)
\end{aligned}
$$

First and second order derivatives for transition and emission probabilities are calculated similarly.

Appendix E

Prior Densities for Physical Parameters

The material of this appendix was provided by Steven Spreckley.

We have obtained prior distributions on the physical parameters from the relevant literature. Primary mass M_1 is distributed according to the [Miller and Scalo, 1979] logarithmic stellar mass function:

$$\xi\left(logM_1\right) = A_0 {M_1}^{A_1}. \tag{E.1}$$

where, A_1 is the power law index or slope of the distribution, the values of which are given in table E.1. We have set a lower mass limit of $0.5M_\odot$ (solar masses) and and upper limit of $100M_\odot$. The density is illustrated in Fig. E.1(a).

Mass range	A_1
$0.5 < M_\odot < 1$	-0.4
$1 \leq M_\odot < 10$	-1.5
$10 \leq M_\odot < 100$	-2.3

Table E.1: Power law index A_1.

The mass ratio q obeys the following prior distribution which is a parameterized version of the distribution presented in [Halbwachs et al., 2003]:

$$p(q) = B_1 \exp(-0.5(q-q_1)^2\sigma_1^2) + B_2 \exp(-0.5(q-q_2)^2\sigma_2^2) + B_3 \exp(-0.5(q-q_3)^2\sigma_3^2),$$

where $B_1 = 1.25$, $B_2 = 1.35$, $B_3 = 2.26$, $q_1 = 0.30$, $q_2 = 0.65$, $q_3 = 1.00$, $\sigma_1 = 0.18$, $\sigma_2 = 0.05$,

and $\sigma_3 = 0.10$. The density is illustrated in Fig. E.1(b).

Period ρ of the system, obeys a log-normal distribution based on the periods of the detached eclipsing binaries identified in the ASAS-3 survey [Paczyński et al., 2006]:

$$p(\rho) = \frac{1}{\sqrt{2\pi}\sigma\rho}\exp\left(-\frac{ln^2(\rho/\rho_0)}{2\sigma^2}\right), \tag{E.2}$$

where $\rho_0 = 4.405$, and $\sigma = 0.77$. For this distribution, we have imposed a lower limit on the period of 0.5 days, and an upper limit of 100 days. The density is illustrated in Fig. E.1(c).

The probability density of the eccentricity, e, for systems with periods less than 5 days has been derived from the eccentricities of the eclipsing binary systems in the OGLE-II galactic bulge survey [Devor, 2005] and is given by,

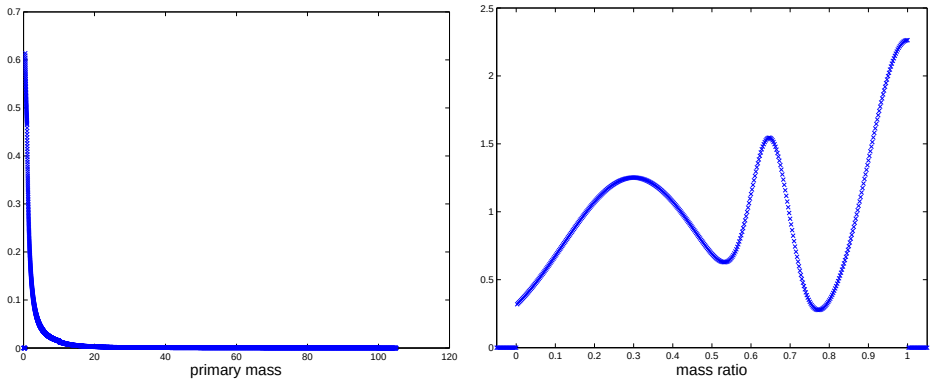
$$p(e|\rho) = C_0\frac{\Gamma}{e^2+\Gamma^2} + D_0\exp(-\frac{e}{2\sigma^2}), \tag{E.3}$$

where $C_0 = 0.18$, $\Gamma = 0.0135$, $D_0 = 0.18$, and $\sigma = is0.26$. This first part of the density is illustrated in Fig. E.1(d). This distribution then transitions into a uniform distribution of eccentricities, but an upper limit to the range of allowed eccentricities is determined by the period over the range of periods from 5 days to 100 days according to:

$$e_{\max} = 0.5\left(1 + \frac{\log_{10}(\rho) - \log_{10}(5)}{\log_{10}(100) - \log_{10}(5)}\right). \tag{E.4}$$

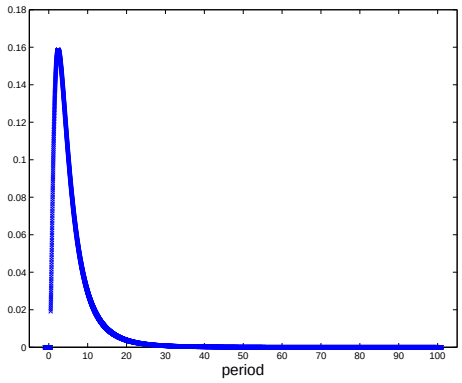
The upper limits for the eccentricities are based on findings by [Halbwachs et al., 2003]. This second part of the density is illustrated in Fig. E.1(e).

The argument of periastron ω is limited to the range $[0, 2\pi]$. We assume that ω is uniformly distributed within its range. Finally inclination $\imath$ is assummed to be also uniformly distributed within its range $[0, \frac{\pi}{2}]$.

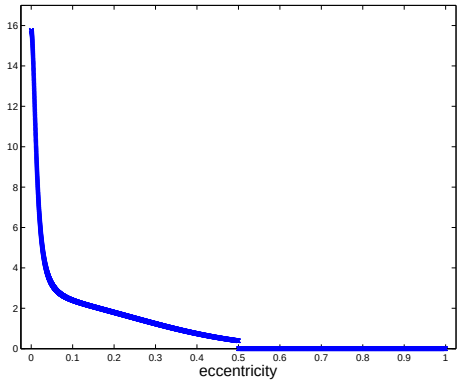


(a)

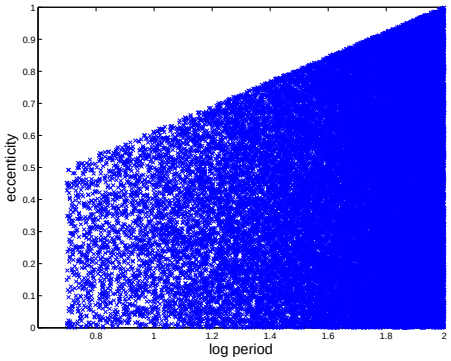
(b)



(c)



(d)



(e)

Fig. E.1: Prior densities on parameters of physical model.

References for Prior Densities

[Devor, 2005] Devor, J. (2005). Solutions for 10,000 eclipsing binaries in the bulge fields of ogle ii using debil. *The Astrophysical Journal*, 628(1):411–425.

[Halbwachs et al., 2003] Halbwachs, J. L., Mayor, M., Udry, S., and Arenou, F. (2003). Multiplicity among solar-type stars. iii. statistical properties of the f7-k binaries with periods up to 10 years. *Astronomy and Astrophysics*, 397:159–175.

[Miller and Scalo, 1979] Miller, G. E. and Scalo, J. M. (1979). The initial mass function and stellar birthrate in the solar neighborhood. *Astrophysical Journal Supplement Series*, 41:513–547.

[Paczyński et al., 2006] Paczyński, B., Szczygieł, D. M., Pilecki, B., and Pojmański, G. (2006). Eclipsing binaries in the All Sky Automated Survey catalogue. *Monthly Notices of the Royal Astronomical Society*, 368:1311–1318.

Appendix F

Disc Areas for Eclipses

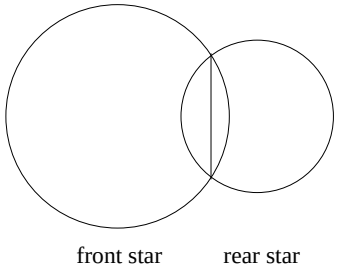


Fig. F.1: Partial eclipse.

In the case where one star is completely obscured by the other, the visible area is of course simply the area of the star in the front. In the case where the disc of the star passing in front of the other is enclosed within the disc of rear star, the visible area of the rear star is given by the difference of the areas of the two discs.

In the case of a partial eclipse (see Fig. F.1), the occulted areas are cut by the line segment through the two points of contact of the discs of the stars. The area cut by the line segment can be easily calculated as follows. The area of a sector of a circle with radius R corresponding to angle θ, as in Fig. F.2(a), is $\pi R^2 \frac{\theta}{2\pi} = R^2 \frac{\theta}{2}$. Area ΔA of the circle cut of by the line segment, can be calculated as the area of the sector minus the area of the triangle with sides the radii of the circle and the line segment. If we bisect angle θ, two equal right-angle triangles are formed. The area of one of them is equal to $\frac{1}{2} R \cos\frac{\theta}{2} R \sin\frac{\theta}{2}$. Thus, the area of the original triangle is $R^2 \cos\frac{\theta}{2} \sin\frac{\theta}{2}$. Area ΔA is equal to the difference:

$$\begin{aligned} \Delta A &= R^2 \frac{\theta}{2} - R^2 \cos\frac{\theta}{2} \sin\frac{\theta}{2} = R^2 (\frac{\theta}{2} - \cos\frac{\theta}{2} \sin\frac{\theta}{2}) \\ &= \frac{1}{2} R^2 (\theta - \sin\theta). \end{aligned} \tag{F.1}$$

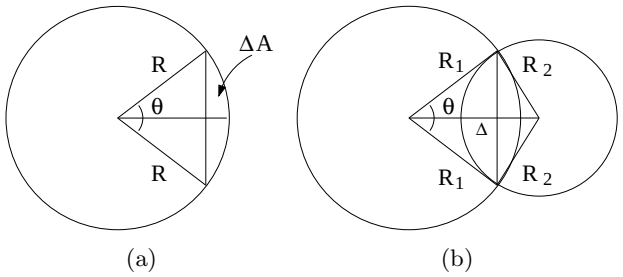


Fig. F.2: Occulted areas.

Considering now both stars, as in Fig. F.2(b), via the law of cosines angle θ is determined:

$$\theta = 2\cos^{-1}\left(\frac{R_1^2 + \Delta^2 - R_2^2}{2R_1\Delta}\right). \tag{F.2}$$

Depending on which star is in front and the state of the eclipse, visible areas are calculated according to table 6.1.

References

[Amari, 1959] Amari, S.-i. (1959). *Differential-Geometrical Methods in Statistics*. Springer-Verlag.

[András, 2002] András, P. (2002). Kernel-kohonen networks. *International Journal of Neural Systems*, 12(2):117–135.

[Bauer and Pawelzik, 1992] Bauer, H.-U. and Pawelzik, K. R. (1992). Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Transactions on Neural Networks*, 3(4):570–579.

[Bilmes, 1997] Bilmes, J. (1997). A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report ICSI-TR-97-021, University of Berkeley.

[Bishop, 1996] Bishop, C. M. (1996). *Neural networks for pattern recognition*. Oxford University Press.

[Bishop, 1999] Bishop, C. M. (1999). Latent variable models. In *Learning in Graphical Models*, pages 371–403. MIT Press.

[Bishop et al., 1996] Bishop, C. M., Svensén, M., and Williams, C. K. I. (1996). GTM: A principled alternative to the self-organizing map. *Lecture Notes in Computer Science*, 1112:165–170.

[Bishop et al., 1997] Bishop, C. M., Svensén, M., and Williams, C. K. I. (1997). Magnification factors for the gtm algorithm. In *Proceedings IEE Fifth International Conference on Artificial Neural Networks*, pages 64–69.

[Bishop et al., 1998] Bishop, C. M., Svensén, M., and Williams, C. K. I. (1998). GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234.

[Bloom, 1979] Bloom, D. M. (1979). *Linear Algebra and Geometry*. Cambridge University Press.

[Brett et al., 2004] Brett, D. R., West, R. G., and Wheatley, P. J. (2004). The automated classification of astronomical light curves using kohonen self-organizing maps. *Monthly Notices of the Royal Astronomical Society*, 353(2):369–376.

[Burden, 1997] Burden, R. L. (1997). *Numerical analysis*. Pacific Grove.

[Cadez et al., 2000] Cadez, I. V., Gaffney, S., and Smyth, P. (2000). A general probabilistic framework for clustering individuals and objects. In *KDD*, pages 140–149.

[Calmet and Calmet, 2005] Calmet, X. and Calmet, J. (2005). Dynamics of the fisher information metric. *Physical Review E*, 71:056109.

[Chappell and Taylor, 1993] Chappell, G. and Taylor, J. (1993). The temporal kohonen map. *Neural Networks*, 6:441–445.

[Cover and Thomas, 1991] Cover, T. M. and Thomas, J. A. (1991). *Elements of information theory*. Wiley-Interscience.

[Crouse et al., 1998] Crouse, M., Nowak, R., and Baraniuk, R. (1998). Wavelet -Based Statistical Signal Processing Using Hidden Markov Models. *IEEE Transactions on Signal Processing*, 46(4):886–902.

[Degroot, 1996] Degroot, M. H. (1996). *Probability and Statistics*. Addison Wesley, 2nd edition.

[Do, 2003] Do, M. N. (2003). Fast approximation of Kullback-Leibler distance for dependence trees and hidden markov models. *Signal Processing Letters, IEEE*, 10(4):115–118.

[Durand and Gonçalvès, 2001] Durand, J.-B. and Gonçalvès, P. (2001). Statistical inference for hidden Markov tree models and application to wavelet trees. Technical Report 4248, INRIA.

[Durand et al., 2004] Durand, J.-B., Gonçalvès, P., and Guedon, Y. (2004). Computational methods for hidden markov tree models-an application to wavelet trees. *Signal Processing, IEEE Transactions on*, 52(9):2552–2560.

[Erwin et al., 1992] Erwin, E., Obermayer, K., and Schulten, K. (1992). Self-organizing maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, 67(1):47–55.

[Falkhausen et al., 1995] Falkhausen, M., Reininger, H., and Wolf, D. (1995). Calculation of distance measures between hidden markov models. In *EUROSPEECH-1995*, pages 1487–1490. ISCA Archive.

[Fine et al., 1998] Fine, S., Singer, Y., and Tishby, N. (1998). The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32(1):41–62.

[Frieden, 1998] Frieden, B. R. (1998). *Physics from Fisher information: a unification*. Cambridge University Press.

[Geusebroek et al., 2005] Geusebroek, J. M., Burghouts, G. J., and Smeulders, A. W. M. (2005). The Amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112.

[Goodstein and Goodstein, 1996] Goodstein, D. L. and Goodstein, J. R. (1996). *Feynman's Lost Lecture: The Motion of Planets Around the Sun*. Norton, New York.

[Gray, 1984] Gray, R. M. (1984). Vector quantization. *IEEE ASSP Magazine*, pages 4–29.

[Guinan and Engle, 2006] Guinan, E. F. and Engle, S. G. (2006). The brave new world of binary star studies. *Astrophysics Space Science*, 304:5–11.

[Günter and Bunke, 2002] Günter, S. and Bunke, H. (2002). Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters*, 23(4):405–417.

[Hagenbuchner et al., 2003] Hagenbuchner, M., Sperduti, A., and Tsoi, A. C. (2003). A self-organizing map for adaptive processing of structured data. *Neural Networks, IEEE Transactions on*, 14(3):491–505.

[Hagenbuchner and Tsoi, 1999] Hagenbuchner, M. and Tsoi, A. (1999). The traffic policeman benchmark. In Verleysen, M., editor, *European Symposium on Artificial Neural Networks*, pages pp. 63–68. D-Facto.

[Hammer et al., 2004] Hammer, B., Micheli, A., Strickert, M., and Sperduti, A. (2004). A general framework for unsupervised processing of structured data. *Neurocomputing*, 57:3–35.

[Heskes, 1999] Heskes, T. (1999). Energy functions for self-organizing maps. In Oja, S. and Kaski, E., editors, *Kohonen Maps*, pages 303–315. Elsevier, Amsterdam.

[Hilditch, 2001] Hilditch, R. W. (2001). *An introduction to close binary stars*. Cambridge University Press.

[Hofmann, 2000] Hofmann, T. (2000). Probmap - A probabilistic approach for mapping large document collections. *Intelligent Data Analysis*, 4(2):149–164.

[Hollmén et al., 1999] Hollmén, J., Tresp, V., and Simula, O. (1999). Self-organizing map for clustering probabilistic models. In *ICANN99. Ninth International Conference on Artificial Neural Networks*, volume 2, pages 946–951.

[Kabán and Girolami, 2001] Kabán, A. and Girolami, M. (2001). A combined latent class and trait model for the analysis and visualization of discrete data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):859–872.

[Kaski et al., 1998] Kaski, S., Kangas, J., and Kohonen, T. (1998). Bibliography of self-organizing map (SOM) papers: 1981–1997. *Neural Computing Surveys*, 1:102–350.

[Kaski et al., 2001] Kaski, S., Sinkkonen, J., and Peltonen, J. (2001). Bankruptcy analysis with self-organizing maps in learning metrics. *IEEE Transactions on Neural Networks*, 12:936–947.

[Kay, 1993] Kay, S. M. (1993). *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Inc.

[Keim and Ward, 2002] Keim, D. and Ward, M. (2002). *Visual Data Mining Techniques*, pages 403–427. Springer-Verlag, 2 edition.

[Kleiberg et al., 2001] Kleiberg, E., van de Wetering, H., and van Wijk, J. J. (2001). Botanical visualization of huge hierarchies. In *IEEE Symposium on Information Visualization, INFOVIS*, pages 87–94.

[Kohonen, 1990] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

[Kullback, 1959] Kullback, S. (1959). *Information theory and statistics*. Wiley, New York, NY.

[Lystig and Hughes, 2002] Lystig, T. C. and Hughes, J. P. (2002). Exact computation of the observed information matrix for hidden Markov models. *Journal of Computational and Graphical Statistics*, 11(3):678–689.

[Manning and Schtze, 1999] Manning, C. and Schtze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

[Minka, 1998] Minka, T. (1998). Expectation-maximization as lower bound maximization. Published on the WWW, http://research.microsoft.com/ minka/papers/em.html.

[Moller, 1993] Moller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533.

[Myung and Daniel, 2005] Myung, I. J. and Daniel, J. N. (2005). Information matrix. In Howel, B. E. D., editor, *Encyclopedia of Statistics in Behavioral Science*, volume 2, pages 923–924. Wiley.

[Ng et al., 2004] Ng, S., Krishnan, T., and McLachlan, G. (2004). The em algorithm. In Gentle, J., Hardle, W., and Mori, Y., editors, *Handbook of Computational Statistics*, volume 1, pages 137–168. Springer-Verlag.

[Oja et al., 2003] Oja, M., Kaski, S., and Kohonen, T. (2003). Bibliography of self-organizing map (SOM) papers: 1998–2001. *Neural Computing Surveys*, 3:1–156.

[Park and Sandberg, 1991] Park, J. and Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257.

[Petersen and Pedersen, 2006] Petersen, K. B. and Pedersen, M. S. (2006). The matrix cookbook. Published on the WWW, Version 20051003.

[Pölzlbauer, 2004] Pölzlbauer, G. (2004). Survey and comparison of quality measures for self-organizing maps. In Paralič, J., Pölzlbauer, G., and Rauber, A., editors, *Proceedings of the Fifth Workshop on Data Analysis (WDA'04)*, pages 67–82. Elfa Academic Press.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Rowe and Hidović, 2004] Rowe, J. E. and Hidović, D. (2004). An evolution strategy using a continuous version of the gray-code neighbourhood distribution. In *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 725–736. Springer-Verlag.

[Salakhutdinov et al., 2003] Salakhutdinov, R., Roweis, S. T., and Ghahramani, Z. (2003). Optimization with em and expectation-conjugate-gradient. In Fawcett, T. and Mishra, N., editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 672–679. AAAI Press.

[Samet, 1990] Samet, H. (1990). *The design and analysis of spatial data structures.* Addison Wesley.

[Singer and Warmuth, 1998] Singer, Y. and Warmuth, M. K. (1998). Batch and on-line parameter estimation of gaussian mixtures based on the joint entropy. In Kearns, M. J., Solla, S. A., and Cohn, D. A., editors, *NIPS*, pages 578–584. The MIT Press.

[Small, 1996] Small, C. (1996). *The statistical theory of shape*. Springer-Verlag.

[Smyth, 1997] Smyth, P. (1997). Clustering sequences with hidden markov models. In Mozer, M., Jordan, M. I., and Petsche, T., editors, *NIPS*, pages 648–654. MIT Press.

[Smyth, 1999] Smyth, P. (1999). Probabilistic model-based clustering of multivariate and sequential data. In Heckerman, D. and Whittaker, J., editors, *Seventh International Workshop on AI and Statistics*, pages 299–304. Morgan Kaufmann.

[Strickert and Hammer, 2005] Strickert, M. and Hammer, B. (2005). Merge SOM for temporal data. *Neurocomputing*, 64:39–71.

[Svensén, 1998] Svensén, M. (1998). *GTM: The Generative Topographic Mapping*. PhD thesis, Aston University, UK.

[Tiňo et al., 2004] Tiňo, P., Kaban, A., and Sun, Y. (2004). A generative probabilistic approach to visualizing sets of symbolic sequences. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–706. ACM Press.

[Tiňo and Nabney, 2002] Tiňo, P. and Nabney, I. T. (2002). Hierarchical GTM: Constructing localized nonlinear projection manifolds in a principled way. *IEEE Trans. Pattern Analysis Machine Intelligence*, 24(5):639–656.

[Varsta et al., 1997] Varsta, M., Heikkonen, J., and del R. Millan, J. (1997). Context learning with the self organizing map. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland*, pages 197–202. Helsinki University of Technology, Neural Networks Research Centre.

[Venna and Kaski, 2001] Venna, J. and Kaski, S. (2001). Neighborhood preservation in nonlinear projection methods: An experimental study. *Lecture Notes in Computer Science*, 2130:485–491.

[Verbeek et al., 2005] Verbeek, J. J., Vlassis, N. A., and Kröse, B. J. A. (2005). Self-organizing mixture models. *Neurocomputing*, 63:99–123.

[Voegtlin, 2002] Voegtlin, T. (2002). Recursive self-organizing maps. *Neural Networks*, 15(8-9):979–991.